Hyper-space of danger and its application to software risks.

Hyper-space of danger has demonstrated its use in the analysis and search for dysfunctions and catastrophic failures, whether industrial or not. Its application to software risks allows to demonstrate that such risks can be processed in a similar systematical approach.
This presentation focuses on the aspects of risk classification and causality determination.

Abstract

After proposing examples of the DSC[1] in the software houses' world, we will see how these CSDs can be adapted for use with the HyperSpace of Danger. Then we will see how software risk reduction and quality improvement can benefit from methods, norms and techniques used by software test engineers. To conclude we will propose different methods to implement these techniques, while taking into account the economic realities of businesses.

Introduction

Information Technology is often considered by laymen as was Alchemy many centuries ago : an obscure science, carried out in remote locations in the company by (pseudo) scientists using words utterly unknown by non-practitioners (WYSIWYG, C#, java, bluetooth, interface, …). Promising a lot, actual results were often not at the level of investments granted. The search for the Philosopher Stone (the one that turns lead into gold) is no different from the frenzy that took a lot of our contemporaries during the Internet bubble.
When Return on Investment becomes a company's prime goal, reaching objectives in a timely manner and within budget, is a priority. Any event that can have an impact on these goals becomes a risk and can not anymore be considered as minor.
Cindynic methods have demonstrated their applicability in different domains (detection of industrial risks) ; They can also be used in the area of software risks.

Definitions

A software project is a group of administrative and technical tasks whose objective is to reach a goal defined by its hierarchy. To be considered a software project, the technical tasks should – in whole or in part – rely on IT techniques such as programming and/or software development. A risk on a project is any action that has an impact on its fulfillment, be it by lengthening its duration or raising its costs. Thus any delay or cost increase for a project can be considered a risk, even if it does not directly apply to technical tasks.
Likewise we can expand the range to include any use of software product. A software product is part of a production chain, just like an engine or a drill. Faulty software need to be assessed just as an engine or a drill has to be examined.
We are thus presented with a paradigm where the use of an IT tool as well as its conception define the area of software risks : a project that takes more time or resources than was planned in order to be completed has suffered a risk ; if it is a software development project or if it uses software tools, it will be considered a software project and the risks will be software risks.

Risk segregation

In the paradigm, it is necessary to distinguish two aspects of the problem :

- The use of a software tool to undertake a task,
- The design and construction of a software product.

The first case is similar to the use of a tool to execute a task : the tool should be adapted to the task. The difficulty arises from the complexity of the task at hand and the tool's specifications are more complex. Lack of development norms and specifications does not make this more easy. The software tool is often designed in such way that all the users of this tool, whatever their task, are connected to the same tool, in fact using the same tool, as if all the tools of a garage were made into one, that could be used simultaneously by all the mechanics. It is clear that if the tool is faulty all users are affected, which is not true if different size (and/or brand) ot screwdriver or spanner (for example) are used : in case of defect (loss, destruction,…) another similar equipment can be used as replacement.
The second case is similar to the creation of civil engineering structures (building, harbor, highway interchange, …) where different subcontractors work together. Nobody in its right mind would build a house without first drawing a plan, architects and civil engineers know the – minimal – standards for the angle of evacuation pipes, norms exist pertaining the number and size of safety exits for public offices. Nowadays, software development is more an art[2] than a science[3]. Standards, norms and methods are applied in a non-

[1] Déficits Structurels Cindynogènes : Cindynogenic Structural Deficiencies, structural deficits liable to increase danger
[2] Art : craft, skill at doing something. (source : Larousse dictionary).

systematic manner and their interpretation varies according to the practitioners. The IT risks in such cases are risks of incorrect or unsatisfactory functionalities generating a defective software application, and risks of time or budgetary overruns in order to reach a satisfactory product..

Examples

In order to understand the use of the ten Cindynogenic Structural Deficiencies (DSC) and apply them to practical cases of IT risks, let's take some examples, each associated to a particular DSC. These DSCs are often seen as cultural problems within the industry. The examples provided are taken from live experiences from French software houses :

DSC1 : Infallibility culture :

In order to convince a prospective customer, the marketing person or the project manager declare that their software house is totally capable to solve the customer's problem, that they already did this for other customer (showing a list of major – well known – companies to convince their prospective customer). Of course « we have the adequate experts and/or specialists ». Back in their software house it is decided to train (most often through self-training) one or more engineer in order to make them appear « expert » or « specialist » to their customer. As if it were possible to create a specialist in a few days …

DSC2 : Simplism culture :

This is an often heard explanation from different managers : « it's simple, make a prototype and adapt it afterwards ». In fact this comes down to simulate a single-user single-location process and then to enlarge its database size and the number of users. This is when one notices problems of concurrent (simultaneous) access to a single record (should one warn that the record is already accessed by another user ?, who has priority when storing data?) backup problems (backup prohibited if there is still a user connected to the database, but the users are all over the country), security problems etc that were not initially planned for.

DSC3 : Non-communication culture :

During a major project involving multiple systems and multiple (internal and sub-contracting) teams, it often happened that the teams did not communicate. This resulted in exchange of data with incompatible formats (ASCII[4] vs EBCDIC[5]).

DSC4 : Navel-gazing culture :

In one of NASA's project (Mars exploration rocket), the guidance programs were being developed in Europe (using the metric system Km/sec) on one hand and on the other hand in the US (using measurement in Miles/sec). Neither one nor the other team had taken into account the possibility of different units, each taking the one they were used to. The result was the loss of the mission.

DSC5 : Dependency of the risks to the production :

Many software houses provide or deliver to the market, on purpose and knowingly, software of dubious quality. This is done in order to satisfy delivery dates, often associated with late delivery penalties. As the level of quality is not specified, they provide patches and other corrections till the quality level is reached or the customer budget is exhausted (whichever comes first).

DSC6 : Dilution of responsibilities :

The multiplicity of actors and the differences in technical competencies between the different actors often results in a dilution of responsibilities : technicians shift the blame to other technicians, or to unclear specifications, or to sub-contractors (or co-contractors), even to products they – themselves – proposed earlier (cf. project Socrate at the SNCF, the French railway system).

DSC7 : Lack of practical feedback :

In software development, it is rare that a software house designs a similar software more than once. The information feedback is thus reduced. What is more the renewal rate of the software development teams (on average 30% yearly) implies that the team is completely replaced after about 3 years. This is another hindrance to a valid feedback mechanism. Analysis of the experiences from

[3] Science : coherent grouping of knowledge related to a category of facts, objects or phenomenon following a set of rules and verified by experimental methods (source : Larousse dictionary).

[4] American Standard Code for Information Interchange, coding of characters on 7 bits as proposed by ANSI.

[5] Extended Binary Coded Decimal Interchange Code, coding of characters on 8 bits as proposed by IBM.

other software houses, and other teams is difficult due to a lack of communication, of statistics and the absence of peer-reviewed publications where teams could share their experience freely.

DSC8 : Lack of cindynical methodology :

In most software houses, lack of written management recommendations allows free creative licence (or lack thereof) for each actor in software development. When it concerns securing electronic data, this aspect is often taken into account after the functional aspects of the software application has been properly taken care of. The risks of side effect between two applications are seldom taken care of and appear only when customer call (read complain).
In a bank, a program to restore nightly backup information destroys information support. The operators, following the procedure in case of incorrect termination of the restoration procedure, take the tapes from the previous day (same trouble) then the earlier days (same problem). When they ask the safe-deposit keys in order to use the last remaining tapes the person in charge of the safe asks the OK from the I.T. systems director who, in view of the available data prohibits the delivery of the tapes, thus safeguarding the accounting data for hundred of thousands of customers.

DSC9 : Lack of training to risks :

Training of IT engineers and technicians is totally lacking from courses curriculum, or is only an optional subject in some cases. This is partly due to the lack of detailed and accurate statistics on the causes of software dysfunction.

DSC10 : Lack of planning of Crisis Situations :

When a major problem (problem that can have an impact on the whole project) occurs, only a few managers (directors, marketing) gather together to determine the best strategy. The crisis is (often) only examined thru the impact it has on accounting and bottom line (quarterly) profits, sometimes with regards to continuing commercial relations with the customer. Seldom is there a planning involving all the branches or departments of the company.

A new approach ?

Analysis of the causes of delays or costs overruns for software applications has been the subject of numerous studies, in Europe as well as in northern America. The resulting suggestions by the authors of these studies vary, but can be grouped in 3 areas :

- Providing formal, procedural methods ;
- Setting up more or less well adapted techniques ;
- Using accepted norms and standards.

The use of these solutions varies according to the country's culture, important in the German- and English-based cultures, weak in latin-based cultures. In order to notice this fact, you only have to compare the number of books published on software testing in English and in French.

In a world where exchanges and techniques are the basis for economic success of the available structures (state, companies, individuals), the reduction of risks on software projects should be a priority. The CSD approach and the HyperSpace of Danger proposes a new and promising approach.

Adapting DSC to the HyperSpace of danger

The HyperSpace of Danger (see page 4) allows to group in a graphical way the different causes of risk. We propose below a suggestion to fitting the Cindynogenic Structural Deficiencies (DSC) to the different axes or dimensions of this five dimension HyperSpace,

1. Statistics dimension : related to the data available from historic events and statistical data DSC : 7
2. Epistemic dimension : related to representations and models created from the facts
   DSC : 1, 3, 8
3. Finality dimension : related to the objectives and goals (the social components of the company
   DSC : 2, 3, 4, 5
4. Deontological dimension : related to law, norms, rules and standards, mandatory or freely accepted
   DSC : 3, 6, 8
5. Axiological dimension : related to the value system of the referent
   DSC : 5, 6, 9, 10

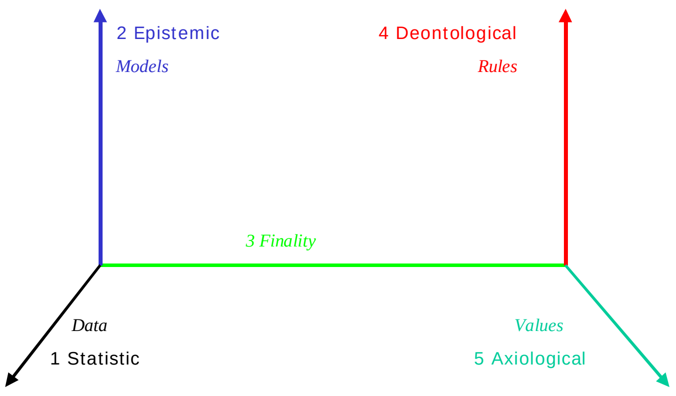

Figure 1 : HyperSpace of Danger

Reducing risks and enhancing quality of software

Different methods and techniques are used in order to enhance software quality :

- Methods for detecting defects already present in the software
  These methods serve to detect defects that were (inadvertently) introduced in the software. They are methods located after coding and potentially on the critical path of the future delivery (time to market).
  - Functional Tests (automated or not)
  - Performance Tests (usually automated)
  - Acceptance Tests (usually manual),
  - Black or White box Tests,
  - Regression Tests, Coverage Tests
- Methods for reducing future defects
  Theses methods and techniques are used to prevent future defects, either by working on present (detected) defects (FMEA, configuration management, defect tracking) or by looking for incoherencies in specifications or in the code before code release (V&V, code review, document review)
  - FMEA (Failure Mode and Effect Analysis)
  - V&V (Verification & Validation)
  - Code and document review
  - Configuration management
  - Defect tracking
- Norms
  Different norms exist to formalize software development and provide reference documents for coding. These norms, methods or standard can be used in software development projects to reduce software risks. While it is certain that using norms and standards will create an initial delay in the project (understanding and working with the norm), positive results are – quite soon – felt.
  Here are some (non exhaustive list of) norms that could be applied to different aspects of software quality :
  - ISO[6] 9126 : Information technology -- Software product evaluation -- Quality characteristics and guidelines for their use
  - ISO/IEC 14598 : Information technology -- Software product evaluation -- Part 1: General overview
  - IEEE[7] 829 : Standard for Software Test Documentation
  - IEEE 730 : Standard for Software Quality Assurance Plans
  - IEEE 1058 : Standard for Software Project Management Plans
  - IEEE 1059 : Guide for Software Verification and Validation Plans
  - ANSI[8]/IEEE 1042 : Guide to Software Configuration Management
  - SW-CMM : Capability Maturity Model for software (CMU/SEI[9])

[6] International Standard Organisation
[7] Institute of Electrical and Electronical Engineers
[8] American National Standard Institute
[9] Carnegie Mellon University / Software Engineering Institute

Any software quality enhancement policy should be implemented using the DSCs.

Statistics dimension :

Different studies, mostly north American studies, demonstrated the impact of lack of quality during the software life cycle and development process. As an average, the cost of a software fix is about 200 times more expensive if it happens at system delivery instead of at system initial conception[10]. 75% of software projects do not end within time and/or estimated budget[11].
The lack of national statistics should not be a pretext to pretend that local (ie national) software engineers are better ; there is no reason to believe this, specially if one takes into account the small number of non US software available on the market.

Epistemic dimension :

Come models exist to determine the density (rate per number of lines) of defects per thousands lines of code (KLOC) ; other mathematical models are available to determine the development work load[12]. The use of these models is often waived and other personal, empirical evaluations are used by the engineer, said results being reduced by marketing or business development personal to take into account the customer budgets. Scope reduction is not envisaged as it would not fit with the customer's requirements.

Finality dimension :

The large number of actors implicated in the definition of the application's finality (goals) makes its definition a risky proposition. On one hand we have the users of the software product, on the other hand we have economical aspects of software development by software houses. The goal for software users is to ease their productivity. For the software houses the goal is that the customer pays what's invoiced. The finality are thus totally different, if not directly opposed. For the economic aspect, the finality is to produce software of a good enough quality, within budget and time constraints. From the user's perspective, the functional aspects and productivity enhancements are most important.

Deontological dimension :

We have seen that a large number of rules, norms and standards have been developed by recognized international bodies (IEEE, ISO, ANSI), however their use is not so widespread. Training in their use is somewhat confidential and their impact on the software development profession is very limited. That their use remains confidential is thus understandable.
What's more, beyond the standards mentioned above, there are a certain number of rules, written or not, determining the way software components seen by the users interact : buttons, drop down lists, etc have default appearances and ways to function (which might slightly vary depending on the operating system PC or Mac), This is also applicable for other components such as screens or web pages.

Axiological dimension :

The system of values pertaining to software applications is difficult to determine. It could be determined by the enhanced productivity of the users (less time to execute a task), it could be determined by the cost of the software or by the – subjective – value the users places on valid data. For similar costs of software development, the perceived cost might be different whether it is a patient tracking software in an intensive care unit, a purchase/sales software in a trading house or a video game.

Risk Reduction Implementation

Implementation can come from two major origins :

- It can be mandatory,
  - Government decrees or laws (such as mandatory software certification of MSSS[13] software by the CRIM[14] in Québec – Canada),
  - Industrial or economical imperatives (cf "Certified for Microsoft Windows " logo and associated software testing)
  - Mandated by the company's management (cf. ISO9000 certification)
  - Mandated by the customers themselves (cf. DoD requirement of CMM compliance for software providers).
- It could be a common practice in the future by becoming part of standard curriculum to engineers and managers alike.

[10] Source : IBM Research
[11] Source : IDC
[12] Function points methods, COCOMO and COCOMO II are such methods
[13] Ministère de la Santé et des Services Sociaux (province du Québec) Ministry of Health and Social Services (Quebec)
[14] CRIM : Centre de Recherche Informatique de Montréal (Québec) – Software Research Center of Montreal (Quebec)

If the software testing profession has known an important expansion in the last decade in northern america, it could be because litigation by customers of software firms when provided with faulty software, and also because strict standards and norms to be adhered to were set up by regulation bodies (ie FDA[15] et DoD[16]). Another, non trivial, aspect is the impact and militancy of engineering associations (ie IEEE, ACM) in the use and spreading of quality methods, norms and standards.

Without a strong mandate from governments, leading economy executives and customers, including end consumers, a strong increase in overall software quality will not exist.
Improvement methods exist, they must become a standard practice within the software development industry.

Suggestions

The approach to software risk reduction should be made thru different channels :

- Raising the awareness of managers and executives, from software houses and customers
- Training future software engineers and technicians, and re-training existing engineers
- Setting up by the government of mandatory software certifications requiring the implementation of cindynic related methods for some software applications such as those related to health, banking, data interchange and/or any software processing data of confidential/personal nature.

Raising the awareness of managers and IT directors is one way to spread information, but should not be the only one, specially since these persons are faced with economic issues and requests from all sides.
Training of future IT technicians and engineers (and future managers) should include large aspects of the security and risk reduction curriculum, by focusing on the personal responsibility of each person. This could ensure better security awareness in future years. Raising the level of present day engineers (ie thru company incentives) will not provide short term results.
Having the government set up a minimal level of certification (i.e. for specific domains such as health, finance, insurance, and/or any automated processing of personal data), in order to have cindynical methods made mandatory could provide impact of these methods to other sectors of the economy. This could serve as reference for other sectors of the economy.
Canada has already determined mandatory certification of software that use their RTSS[17] network, the United States through their FDA and DoD have defined norms and methods to enhance quality of software, other countries have already or will start similar improvement processes. The IEC[18], an independent and impartial body could (should ?) be at the forefront of these efforts, proposing seminars to inform managers and executives on one hand and by lobbying elected officials and training institutions to train the 21st century IT personnel on Risk Reduction

Author

Bernard Homès M. Sc., a member (Belgian) of the IEC is consultant, international speaker and specialist in software testing. He is also a member of the French chapter of the IEEE (Assistant VP) and participates in the Advisory Boards of different conferences on software quality, as well as belonging to the workgroup on updating the IEEE Std 1074-1997 (Standard for Developing Software Life Cycle Processes) and IEEE Std 829 (Standard for software Test Documentation).
Bernard Homès can be reached at bhomes@tesscogroup.com and at bhomes@ieee.org

[15] Food and Drug Administration
[16] Departement of Defence, USA.
[17] RTSS : Réseau de Télécommunications Socio-Sanitaire : Social and health telecom network.
[18] IEC : Institut Européen de Cindynique, European Institute for Cindynics

Deficits Systemiques Cindynogenerated Table

Hereunder a summary table for the ten DSCs.

	Num	Designation	Classical symptoms	Application to Software
CULTURAL ORGANISM	DSC 1	Infallibility culture	We are certain to succeed. This system is guaranteed agains any defects.	You will have your software delivered in n days. We have the experts, it will fulfill your requirements.
	DSC 2	Simplicity culture	Our problem is not complex, we reject the idea of a system, it works without complex methods.	The proposed solution is the simplest and we are experienced with such applications.
	DSC 3	Non communication culture	One can not live by questioning certain evident truth of our trade, Our company's hierarchy does not agree that technical practices could be questioned. We seldom discuss of practical operations. The personnel speaks Hindi, the crew is portugese and the passengers are Norwegians.	Our methodology has been proven in many cases. The technological evolutions are followed and set up ; our software engineers are used when needed (no coordination ?)
	DSC 4	Egocentrical (inward-looking) culture	We are leaders and save a large amount of time by not looking elsewhere what's being done. We have always been first in perceiving the problems in our profession. We are certain that our competitors are lagging behind in terms of security.	We are experienced, thus we don't need standards or norms to tell us how to work. « we have a proven track record [...] in the principal sectors that are ours [and] an in-depth knowledge of the professions and economic challenges of our customers [19] »
ORGANISATION	DSC 5	Subordination of the risk management functions to the production functions or other risk generating functions.	The security responsible is a collaborator like another from the one responsible for the production. We shall not reduce the prerogatives from the production manager or complexify his/her task. We are burdened by staff, it is not now that we should invent another OK, there are risks, but this should not be a reason to scramble our structures.	We shall include the security aspects after the functional aspects are developed. Security will consist of a password, … (nothing specifies what should happen in case of invalid password, nor how the data will be kept confidential if they are accessible thru databases, nor how the data transfer would be protected…)
	DSC 6	Dilution of responsibility No precise description of risk management tasks. Lack of designated responsible for the tasks.	We have rejected any formalism in our organization, anyone can speak freely. People are adults and know perfectly well what they have to do without being told or reminded of it.	When defining the link between two applications, none of the participants mention the risks associated to the loss of the link (loss of data, no restart points, data duplication, …)
MANAGEMENT METHOD	DSC 7	Lack of feedback on experience.	Practices and processes deemed dangerous in other organizations are kept current. No attention is given to precursor signs appearing in the same profession. No systematic exploitation of facts about worldwide dysfunctions in the same technical domain.	Technology watch, when present in the software houses is geared to new software, not towards dysfunctions in similar domains. Moreover, dysfunctions are often not publicized nor elaborated upon in order to protect the corporate image.
	DSC8	Lack of cindynic method in the organization.	In this sector, one should recognize that there was no manual or written instructions from management or executives.	A large number of companies do not have methods to analyze and/or reduce software riks.
	DSC 9	Lack of training in cindynics adapted to every personnel category.	The production personnel were cought unaware and made errors that aggravated the problem.	The procedures set up are often ambiguous and lead to situations where risks are enhanced.
	DSC 10	Lack of crisis situation planning	When we heard this horrible noise, everybody started running in every direction.	The only point taken into account in crisis situations are those of data loss. During the September 11 events, the risks were not limited to the data, but also included material, human and functional aspects, also for customers of the victims in the twin towers of WTC.

[19] These mutually exclusive assertions are present on the same web page from a known software house