The Five Levels of Conceptual Maturity for IT Teams
History of the Model
A Model of Conceptual Maturity
The Levels of Conceptual Maturity
The Usual Maturity Path
A Maturity Model Related to Complexity
What is the Required Level of Maturity for my Project?
IT Industry Should Target Maturity

The Five Levels of Conceptual Maturity for IT Teams

IT is a strange activity because it is an industry that has troubles getting out of a craft industry approach. Constantly moving to new exciting technologies and constantly influenced by massive technology centered marketing, IT people stay durably too technical and have recurring troubles to achieve IT projects.

History of the Model

Years ago, I was consulting a CEO and was trying to explain to him why his teams were failing big IT projects. A lot of executives today are digitally aware in their daily business practices but, when it comes to organizing new IT projects inside their company, they frequently discover that there is a huge gap between what they wanted to achieve and what is really done. Money is not the problem, and many big companies invest massive amounts of money in IT projects without getting the most of it. The fact is, when you invest too much in an IT project, your chances of success decrease drastically.

I came to the CEO with this maturity model. I don't proclaim it to be perfect, far from it: it's just a representation of the problem. But the model is quite useful to explain and diagnose the capability of teams to perform IT projects and the levels of complexity attached to those projects.

A Model of Conceptual Maturity

IT is an abstract discipline manipulating abstract concepts. In IT, it is always possible to "make the code work", even if the cost and the delay to do that are unacceptable.

The problem of IT is that you can pay the price of bad choices years (even decades) after you did the investment - and you pay each year, possibly more and more year after year for the exact same modification. Bad IT choices imply increasing the technical debt and jeopardize the capability of achieving future projects (at least you will continuously increase their cost). In that sense, IT is quite near from traditional industry.

The fact is, in tradition industry, people are specialized in jobs: not every automotive worker will design the brakes or the engine. In IT, the situation is blurrier. Most IT people don't know what they are capable of doing, because specialization of jobs has not yet reached an industry agreement, and because it seems always possible to "make the code work" (despite of any consideration about quality/evolutions/money/timing/etc.).

However, in IT, there are several levels of concerns that require different skills and different conceptual approaches. Over my 20 years in the IT business, I identified five levels in this conceptual world of software systems creation. Generally, in their professional life, people climb the rungs of the ladder in the same order, which makes the model a "kind of" maturity model.

The Levels of Conceptual Maturity

I will describe briefly the most important aspects of the various levels. I don't pretend to be exhaustive and I use more detailed materials and samples when I play the consultant for executives.

Level 1: Development

Most IT people know how to code, how to produce programs. For that purpose, IT people use programming languages (like Java, Cobol, C++, C, etc.). Behind the coding level, we can add the mastering of the development environment: development standards, debugging, versionning and branching management, build management, testing management, etc.. All those skills are crucial to good developments. Considering all that, if we rated people from 0 to 1 on this topic, I would say that most developers in the market are between 0.5 and 0.8 in development.

Level 2: Design

IT uses object orientation programming (OOP) commonly for more than two decades. However, OOP is conceptually complicated. To be explained to the non-IT people, design is the activity of building a set of programs structure and knowing where to put the code inside the program structure. Depending on how the programs are internally structured, you will face more or less troubles in the future maintenance and evolutions. Design activity uses lots of design patterns (level 2) which are generally badly mastered by IT teams because IT teams think too often at the "code level " (level 1). With years, and with Agile methodologies, IT people were pushed back to level 1. Design is complicated. IT people have to think before they code (very hard for developers); they have to forecast what will be the maintenance. Few people are really capable of good designs.

Level 3: Application Architecture

Architecting an application today is not easy, because they are a lot of trendy "off-the-shelf" components, a lot of competing "sexy" technologies. The hype around technologies is extreme and IT people can change frameworks very quickly to follow the last trend. Architecting an application is the skill to choose what components (reusable or not) will be used into an application. It enables the application to be developed correctly by several people, to isolate "modules", to manage customizable dependencies inside the IT systems. This is not an easy job. The objective is not just to realize software but to be able to maintain them on the long run and whatever their functional perimeter (that can become quite large with years). In order to build good application architecture, you have to think about "component borders" and be able to estimate the pros and cons of using a trendy component in the market. That's quite a hard job.

Level 4: IT Architecture

IT is everywhere and companies, even SMEs, have quite quickly a lot of IT systems. Being able to make those systems communicate together without being too hardly coupled, being able to make the IT systems as a whole evolving in a reasonable timing for reasonable costs, being able to communicate with the outside of the company in a safe manner, all that is part of IT architecture. In IT architecture, you also manage the positioning of features inside softwares: when a function is implemented in the wrong software, you can loose millions of dollars. In IT architecture problems are also the integration problems: a good IT architecture helps decreasing the QA costs and improving the TTM of the projects. Good IT architects are very rare in the market. It seems a pity because, for large companies, IT systems are one of the biggest problems, for instance in M&A phases. To add to the complexity, the IT market sells a lot of "off-the-shelf solutions" in that area, solutions that often bring confusion in IT systems, when put in place by non level 4 people.

Level 5: Enterprise Architecture

The term "Enterprise Architecture" (EA) is at the center of a lot of discussions because the industry is not clear about what an enterprise architect is supposed to do. I will provide my own pragmatic vision (which can be challenged). An enterprise architect must primarily ensure that company strategic intentions are concretely declined into the IT systems through auditable projects and programs. Secondly he/she must ensure that the changes brought to the softwares can be managed by end users, that software changes are not destroying efficient business processes but improving them. This is quite a work that requires a lot of IT skills but also a good business understanding and enhanced communication skills.

The Usual Maturity Path

One common path for IT-related designers or architects is to go from level 1 to level 5. Each level is requiring, following my experience, in average, at least 2 years of real life practice. Most often, when you work at a certain level, you have to consider also the impacts of what you are doing in all other lowest levels (that's why enterprise architects that never coded often make me uncomfortable, because they cannot imagine the consequences of their decisions).

Most people will find happiness in a certain layer and will excel in that layer because it fits their mind and taste and it is perfectly adapted to the level of complexity they like to manage. Without a proper project experience and without training and/or coaching, it is quite hard for people to change levels without having to work for months in real life at the concerned level.

A Maturity Model Related to Complexity

The model describes the capability to solve IT problems at the proper level, to manage the complexity of the IT problem with the proper approach. For instance, you can take a level 4 problem with a level 1 approach but you have 70% of chances of never solving the problem completely and 95% of chances to spend millions of dollars fixing "things that don't work". The "things that don't work" were created by people because of their problem solving approach (level 1 instead of level 4). Using the proper approach would never have generated those costs and troubles.

What is the Required Level of Maturity for my Project?

Regarding this maturity model, executives should appreciate the level of maturity of their internal IT organization. They are then able to ask themselves the question: what's the adequate level of maturity required to achieve the project I want? Or, we can ask the question the other way round: what projects can I safely achieve considering the maturity of my internal IT organization? Those questions are crucial.

When I look at failed projects, I often see people having a maturity level between 1 and 3 trying to master a multi-million dollars IT projects requiring a level 5. Integrating external consultants can do the job but not in all cases. If your business is complex and quite specific, it is preferable to bet on internal skills in the long term for IT people to understand your business in depth. Moreover, who chooses the casting of your external consultants? Your internal IT organization, and few organizations are able to hire consultants that know more than they do.

This is exactly what happened when big companies got out of the mainframe. Mainframe teams were mostly level 1, some of them were level 2 (level 3 is not relevant in old mainframe application design and level 4 is often just about producing files for the outside or consuming them). Giving such a team an IT program with distributed applications in various technologies to build is doomed to failure, whatever the money and whatever the quality and dedication of people. For sure, with time and the appropriate training, most people can climb rungs in the ladder. But without time and training, people will keep on reproducing failures. Suppose the management is not aware of the core problem and people get fired for project failure, you can end up taking big outsourcing decisions based on a situation that nobody really understood.

IT Industry Should Target Maturity

Because it is only "soft"-ware and not hardware, the IT industry seems to like the level 1 so much that every serious attempt to raise to level 2 or 3 is pushed back to level 1 by the IT providers. IT providers love to focus their customers on the "code level": because they cannot see the bigger picture and buy their products even if they don't need them.

If the IT industry created models for IT project management (like PMP), the increasing complexity of existing systems and new projects require the appropriate level of maturity in the project designers/architects teams.

The problem today, in most companies, is not only to "code" nor only to "manage projects": it is to also to maintain the running systems and make the huge amount of legacy applications evolve without risks and conforming to the company strategic intentions. Most IT systems are a pile of geological layers of outdated technological hypes. The code of today becomes the legacy of tomorrow. If some industries (like aviation) forecast their investments for decades, IT should begin to do the same, to work on the adequate long-term skills that are required to ensure durable and quality IT services.

IT industry should target maturity and maturity comes with the identification of our level of mastering of IT complexity.

(*April 2015*)