Exploring the Use of the Cynefin Framework to Inform Software Development Approach Decisions

Rory V. O'Connor
School of Computing
Dublin City University
Dublin
Ireland
Rory.OConnor@dcu.ie

Marion Lepmets
Regulated Software Research Centre
Dundalk Institute of Technology
Dundalk,
Ireland
Marion.Lepmets@dkit.ie

ABSTRACT

Choosing an appropriate software development process is a complex and challenging task, exacerbated by the fact that all process models require a certain amount of tailoring to fit to the business environment of any specific organization in which the model is to be deployed. This position paper proposes that one of the potentially most significant factors impacting how a team should structure their software development process is domain (contexts defined by the nature of the relationship between cause and effect) the team is in, an approach pioneered by Snowden with The Cynefin Framework. Cynefin (pronounced Ku-nev-in) is a decision framework that recognizes the causal differences that exist between different types of systems and proposes new approaches to decision making in complex social environments and new mechanisms of understanding levels of complexity as decisions are made. It is argued that using the Cynefin framework for classifying important software process selection decisions assists in choosing the right process for the given situational context. This position paper provides an overview of systems thinking and the Cynefin framework that organizations can use to detect the significant characteristics of the domain in which they operate which has a direct and significant affect on the software process approach (model / methodology) chosen.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management - Software process models.

Keywords

Cynefin, software process improvement, process adaptation, process evolution.

1. INTRODUCTION

The motivation for this position paper is based upon a perceived lack of a holistic systems approach to software process decisions, such as model / methodology selection and process improvement selection. It has long been acknowledged that different systems require different decision making approaches and solutions. Traditional management techniques have been developed to deal with Ordered systems, starting with Scientific Management and later Business Process Reengineering and Systems Thinking. Unfortunately these approaches have limited value when applied to Complex environment such as software development. We propose taking a holistic approach, using the Cynefin Framework [1], where an organization is seen as a system, and where process improvement is but a part of a system within the larger organizational system.

It has been observed that when it comes to the software development process, there is no universal process model suited to all situations [2]. The process models require a certain amount of tailoring in order to be applied beneficially to organizations. Most software organizations engage in the tailoring of standard software process models to their own particular operating context such as the size of the company, the target market, and project and system type [3]. The reason for this is that the process models themselves offer a generic solution and therefore require an approach to allow alignment between process goals and the organization's goals and situation. Furthermore, in terms of process improvement, all too often processes are assessed in isolation within an organization. The focus is on the improvement of a single process area without considering its impact on other processes, on the organization's business or product quality goals thus taking too narrow a view.

Best practice for software development is highly dependent on the context of the organization. In software development organizations today classical best practices from lifecycle models and standards are rarely adopted right out the standards but tailored to contribute most to the organization. There are times, though, when adopting best practices might not be feasible– when the context of the organization is changing so quickly that implementing any practice does not make sense because each situation is too different from the previous one [15]. How do organizations know when to adopt best practice frameworks and how much effort is required to tailor them? Does it bring value to adopt software development best practices in one situation but not in another? We aimed to explore the situational factors behind software development approaches, i.e. which approach brings greatest benefits in which software development context.

This paper explores the potential use of Cynefin, a decision framework that recognises the causal differences that exist between different types of systems and proposes new approaches to decision making considerations regarding process adoption and improvement.

2. CYNEFIN

Systems science argues that the only way to fully understand why a problem or element occurs and persists is to understand the parts in relation to the whole [4]. Systems' thinking encourages understanding a system, i.e. any set or group of interdependent or temporally interacting parts, by examining the linkages and interactions between the elements that comprise the entirety of the system. In other words, systems thinking views problems as parts of an overall system, rather than reacting to specific parts,

ICSSP'15, August 24–26, 2015, Tallinn, Estonia
ACM. 978-1-4503-3346-7/15/08...
http://dx.doi.org/10.1145/2785592.2785608

outcomes or events and potentially contributing to development of unintended consequences. Understanding the internal interactions requires integrating the components into something larger and more capable than the components represent alone. In addition, a system resides within a comparable grouping of environmental factors that might also be called a context.

Cynefin was first developed by Dave Snowden in 1999 in the context of knowledge management and organisational strategy. By 2002, it had developed to include complex adaptive systems theory [1]. Essentially Cynefin is a sense-making framework to help make sense of complex systems by explaining behaviours, decision-making and practices in terms of people's patterns of multiple experiences, personal, cultural and business based. The word "Cynefin" is a Welsh word that means "habitat", but more richly includes notions of the multiple experiences that people have in aspects of their lives. These experiences are a complex mixture of the personal, the wider cultural, and the business based or work place based. Cynefin is based on the notion that '*humans use patterns to establish order in the world and make sense of things in complex situations'*. Cynefin originated in the practice of knowledge management with the aim of helping managers to 'break out of old ways of thinking and to consider intractable problems in new ways' [1].

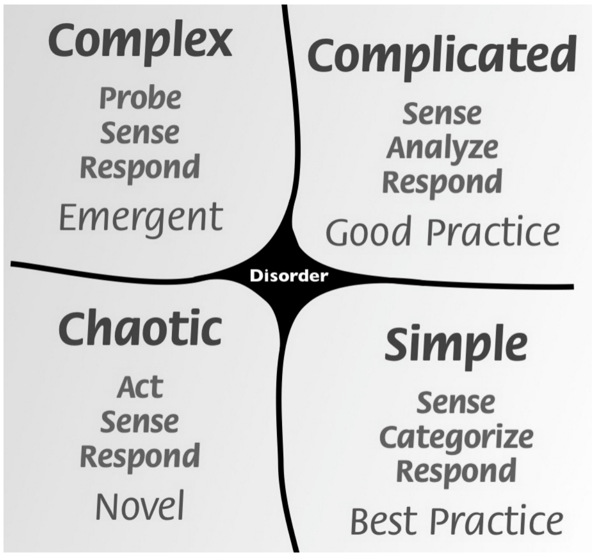

Figure 1. Cynefin framework.

In simplest terms, the Cynefin framework exists to help us realize that all situations are not created equal and to help us understand that different situations require different responses to successfully navigate them.

The Cynefin framework (illustrated in Figure 1) is a phenomenological framework, meaning that it is about how people perceive and make sense of situations in order to make decisions [1]. Cynefin has two large domains: *Order* and *Unorder*, each containing two smaller domains - *Simple* and *Complicated* in the *Ordered* domain, and *Complex* and *Chaotic* in the *Unordered* domain. In the centre of the framework is the fifth domain called *Disorder* where multiple perspectives fight for prominence, factional leaders argue with one another and cacophony rules. Disorder should be avoided by organizations as it disrupts work. In the domain of order, the most important boundary of sense-making is between what we can use immediately (what is known) and what we need to spend time and energy on finding out (what is knowable). In the domain of *Unorder*, distinctions of 'knowability' are less important than distinctions of interaction; that is, distinctions between what we can pattern (what is *Complex*) and what we need to stabilize in order for patterns to emerge (what is *Chaotic*). In the *Ordered* domain, the whole is the sum of the parts and the optimization of the system can be achieved by the optimization of the parts. In the domain of *Unorder*, the whole is never the sum of the parts as any action changes the nature of the system. Cynefin's value as a sense-making framework lies in helping system decision-makers understand where their systems lie among these domains, and by extension, what kinds of tools, approaches, processes, or methods are more likely to work successfully in a given system [4]. The 5 Cynefin domains (as illustrated in Figure 1) can be summarised as follows:

- **Simple** is the domain of best practices, where problems are well understood and a solution requires minimal expertise. Many issues addressed by help desks fall into this category. They are handled via pre-written scripts. The correct approach is to sense the situation, categorize it into a known pattern, and apply a well-known, and potentially scripted, solution.
- **Complicated** is the domain of good practices, where you are likely to know the questions that need to be answered and how to obtain the answers. Assessing the situation requires expert knowledge to determine the appropriate course of action. The correct approach is to sense the problem and apply expert knowledge to assess the situation and determine a course of action.
- **Complex** is the domain of emergent solutions, where there are unknown unknowns and the final solution is only apparent once discovered. The correct approach is to develop and experiment to gather more knowledge to determine the next steps, with the goal of moving your problem into the 'Complicated' domain.
- **Chaotic** is the domain of novel solutions where the immediate priority is containment. The correct approach is to triage, once you have a measure of control, assess the situation and determine next steps, with the goal of moving your problem into another domain.
- **Disorder** is the space in the middle where you don't know where you are and the priority one is to move you to a known domain. The correct approach is to gather more information on what you know or identify what you don't know to be able to move to a more defined domain.

To use the Cynefin framework when trying to categorize problem space, one must inspect the relationship between cause and effect of the problem space. If the relationship between cause and effect is straightforward and obvious to all, then you problem is in the simple domain. If the relationship between cause and effect is not obvious, but can be analysed in advance, then you have a *complicated* problem. On the other hand, if the cause and effect can only be determined with the benefit of hindsight, then you are in the *complex* domain, while if there is no obvious relationship between cause and effect, you are in the *chaotic* domain.

Cynefin's value as a sense-making framework lies in helping system decision-makers understand where their systems lie among these domains, and by extension, what kinds of tools, approaches, processes, or methods are more likely to work successfully in a given system [4].

It is most important to understand that organizations live as whole systems, not as a collection of independent processes. And these systems exist in, and interact with an external environment that includes other systems as well as situational factors that can be irregular, highly variable and unpredictable [4]. A significant number of organizations' situations today qualify as *complex*. Their environment may change in short but irregular, unpredictable cycles, requiring the organization to adapt internally to avoid degradation. Therefore decision-making processes depend on the situation. In a *Simple* situation, decision-makers sense, categorize and respond, i.e. they assess the facts of the situation, categorize them, and then base their response on the established practice [5]. Examples of this abound in standard procedures where all that is necessary is to decide what procedure to follow or to make other minor decisions within the procedure. The way incidents are handled by the Service Desk - received, then categorized and responded to - is an example of such a decision-making process. In the *simple* context there is no analysis of the impact customer satisfaction has on the entire software product or service system.

In the *Complicated* context there are no established best practices that can be applied automatically, with little thought. Instead the decision-makers 'sense and analyze' the facts to understand several options and their consequences on multiple levels, and finally respond. This can be observed in software development during the project planning phase when one or more domain experts consider the various stated or implied project goals and whatever is known about constraints, resources and risks before deciding how the project will be carried out. The decision-making process requires analysis, possibly by domain experts, indicating that the qualitative measures are gathered and analyzed before the decision-makers can respond to them.

In the *Complex* domain, the decision-makers cannot impose a course of action but should allow the path forward to reveal itself while conducting experiments that are safe to fail. In other words, the decision-makers should probe first, then sense and finally respond. This can be observed when an organization forms a cross functional team to investigate and, if possible, derives an innovative solution to some situation that the organization's standard processes are inadequate to address. In the complex system, the decision-makers should constantly observe the environment to understand the dynamic forces around their organization. Here the decision-makers should also understand how the system elements affect the behaviour of the entire system. Because there are no patterns here the best management approach is experimental, usually some form of 'probe and learn'. For example, in the event of a system or service failure the decision-makers need to probe into the system, observe its responses and analyze the cause of the failure from those responses. The possible ways forward will emerge from such an analysis.

3. CYNEFIN AND SOFTWARE PROCESS

3.1 Cynefin and the Software Process Perspective

The *Simple* and *Complicated* Cynefin domains require project leaders to adhere to a more fact-based management style. The *Simple* domain is argued to be the domain of 'best practice' and is characterized by stability of the organisation and a clear cause-and-effect relationship, typically one in which the correct course of action or decision is often self-evident and undisputed, where all parties share an understanding that results in commonly agreed decisions.

The *Complicated* Cynefin domain can be considered to be the domain of 'good practice' where there may be multiple competing appropriate solutions and where a clear relationship between cause and effect can be drawn. This domain requires expertise, investigating multiple options for possible software decisions. Here a project manager must not only listen to the advice of fellow team members but also embrace novel thoughts and solutions from others. This requires a willingness to experiment and often involves more creative approaches to enhance novel thinking and ultimately optimal solutions.

The *Complex* domain is typically the area that causes the most difficulty for process improvement. Many software development issues fall into this category, where tacit knowledge ("Know How") is more important than explicit knowledge ("Know What") and adaptation of processes is necessary for success. In the delicate balance between process adherence and organisational structures, it is in this complex domain that recognition of starting point and appreciation for emergent order is key for positive outcome.

Both *simple* and *complicated* domains are heavily process oriented where the guidance of the process models has potential for the most benefit. While in the *simple* domain, the process model guidance may be considered sufficient to tackle a situation, the complicated domain requires additional goal alignment to maximize benefit from the process model [6].

The *complex* domain presents the biggest challenge for process models. This domain is characterized by synergy of people, open-mindedness and innovativeness in problem solving, and goal internalization in decision-making, which process models do not cover. While agile development methodologies and Scrum project management might be best suited to the *complex* domain in software development so far, there is little in the way of explicit guidance for iterative process improvement [7]. This issue has also not been widely recognized in industry. We suggest that existing process models, as they stand today, are not suited to the complex Cynefin domain. However, the majority of organizations today are operating in *complex* situations. Therefore a significant issue to be addressed is what type of process models can provide a solution and if any amount of tailoring of the existing process models might be enough? So far there have been only few attempts to study dynamic capabilities in software development so this could be a fruitful area for research.

3.2 Deploying Cynefin in a Process Context

The Cynefin framework can be used to explain the orientation of start-ups towards flexible and reactive development approaches. It has been suggested by Paternoster et al. [8] that in the context of Cynefin, software start-ups cross the *complex* and *chaotic* domains. Those two domains represent the areas where applying rigorous process management to control development activities is not effective, because no matter how much time is spent in analysis, it is not possible to identify all the risks or accurately predict what practices are required to develop a product. Instead, flexible and reactive methods, designed to stimulate customer feedback, increase the number of perspectives and solutions available to decision makers. Moving from *complex* to *chaotic* domains, software start-ups open up new possibilities for creation, generating the condition for innovations. Therefore, any process tailored to the start-up context needs at least to allow, but optimally even facilitate movements between *complex* and *chaotic* domains that are intrinsic in the innovation generation of start-ups. In our opinion, this is the main requirement for future attempts of adapting software engineering processes to the start-up context.

Paternoster further contends that developers should have the freedom to choose activities quickly, stop immediately when results are wrong, fix the approach and learn from previous failures [8]. However at some point, in preparation for growth, start-ups need to plan for scalable processes. Similarly to SMEs [9], they need to find a balance between flexibility and repeatability in their organizations' knowledge management and processes.

Pelrine [10] reports on conducting a facilitated awareness sessions of the Cynefin framework with software development professionals where participants related concepts of interpreting their cognitive biases related to software development, and of understanding software development in terms of Cynefin domains. Table 1 offers a sample of typical software development tasks / challenges provided by participants, together with their sense-making results.

Table 1. Typical tasks / challenges mapped to Cynefin domains (adapted from [10])

Domain	Example 1	Example 2
Simple	Knowing when a task is done	Monitoring time spent in phase
Complicated	Fixing a build	Ambitious timeline
Complex	Changing requirements	Task estimation
Chaotic	Retrospectives without consequence	Project scope too large
Disorder	No release deadline	Lack of trust

Software development is a diverse field of endeavour domain, with software development activities representing all of the Cynefin domains and the interactions between these software development activities themselves frequently being of complex nature [10]. Further, as software development can be characterized as a multi-level domain where most activities are composed of a significant number of sub-activities, it is reasonable to suggest that many of these sub-activities may themselves be located in different Cynefin domains.

Pelrine [10] suggests that activities tend to be weighted more to the complicated and complex domains, with activities related to the coding aspect of software development landing in the *complicated* (or sometimes *simple*) domain, and activities associated with project management landing in the *complex* (sometimes *chaotic*) domain. Tasks dealing with interaction with a computer tended to be in the *ordered* domains, tasks dealing with interaction with other humans tended to be in the *unordered*, i.e., *complex* and *chaotic*, domains. Although this does not suggest that the entire software development activity as a whole is complex, it does suggest that many parts of it are amenable to analysis and treatment using complexity-based tools and techniques.

3.3 Harnessing Cynefin to Inform Process Decisions

Cynefin can be used to determine how best approach a team's product development process. The originator of Cynefin refers to this as 'requisite applicability' [5] which simply put means "dependent on which domain you are in, you should think differently, you should analyze differently… rather than one size fits all which has been the tradition of management theory". In support of this viewpoint, Snowden [5] points out the risk that teams are often used to a specific approach and therefore keep following this approach as "we interpret the situation based upon our personal preference for action".

Accordingly we can use Cynefin to recognize the different domains and apply different practices to each domain. For example, as stated in section 2, the *Simple* domain purview of best practices, therefore in this domain one is seeking to constantly improve the process with efficient, effective and repeatable processes. Whereas by comparison in the *Complicated* domain it is the search for good practices and the expertise deployed which should inform decision making. On the other hand in the *Complex* domain it is the search for emergent practices, that must be clarified, evaluated via 'probe and sense'. Finally in the *Chaotic* domain it is more the application of novel practice, as a 'reaction' to the situation where judgement and experience are applied to find original solutions to tackle mostly unique situations in a 'learn as you go' scenario [11].

Therefore we suggest that this means we can infer an appropropiate approache(s) based on which Cynefin domain we are faced with and by extension what software development practices best apply.

To illustrate this point, let us examine the *Simple* domain which is characterised by 'Sense – Categorize – Respond'. In a *Simple* domain because the solution is well known we expect that we already know the majority of the product / market information ahead of time. We can therefore postulate that the Waterfall model is an appropriate choice, where a fixed process of product development is followed with defined sequential steps, no incremental delivery and possibly low levels of iteration or change requests [12].

By contrast the *Complex* domain is characterised by 'Probe – Sense – Respond' and we typically expect that product / market information will be steadily gained over time, often by experiments over a period of time. We can therefore postulate that agile methods such as Scrum would be an appropriate choice to address the work approach of the *Complex* domain the best, as there are 2 underlying key principles to be aware of that support 'Probe – Sense – Respond', which are 'Iteration' (Probe, Sense) and 'Incrementalism' (Respond).

The *Complicated* domain is characterised by 'Sense – Analyse – Respond', where typically several different solutions can be successfully employed and therefore the appropriate response can be situational factors [13]. This is an area where the people and team should be the focus rather than a specific process. Snowden [5] offers the following advice in the *Complicated* domain "*there are several different ways of doing things, all of which are legitimate if you have the right expertise. Trying to force people to adopt one of them can actually be counter-productive*". Essentially you make your choice based on the specific situation and the expertise of the team that you have. Therefore team empowerment maybe consider to be of greater importance and accordingly an agile method with a strong emphasis on team empowerment and ownership would seem suitable.

Finally the *Chaotic* domain is more the application of novel practice, as you are 'reacting' to numerous situations where judgement and experience are applied to find original solutions to tackle mostly unique situations where you are learning as you go [11]. Given the high levals of risk associated with novel practice a more risk adverse lifecycle aproach such as the Spiral model may be considered more suitable.

3.4 Discussion

It is worth noting that there is a risk of always resorting to a default position of treating software development like it is a *Complicated* [domain] problem, where we enter into some analysis in order to come up with a plan to be followed, only to be surprised that the plan does not yield successful software project. Often then with the benefit of hindsight, it can be seen that this did not work as we were dealing with a problem that is in [for example] the *Complex* domain [14].

As software development organisations face *Complex* or *Chaotic* domains they must take on board more new learning, more situational assessment and understanding, looking and combining capabilities to manage emerging patterns and knowledge, applying experiences, looking for diversity of opinions and searching for new wisdom or insights. Here expertise and experience, collaboration and relationships need significantly leveraging, as you often diverge / converge constantly as you work through the potential answers [11]. The mindset here is different and it is one that is based on detection. Innovation is far more demanding, pushing frontiers, exploring discoveries, dealing in a series of exchanges and recognizing emerging patterns to piece together real 'new to the world' innovations.

The Cynefin framework can be used to guide our approach to a set of different situations, but the characteristics also explain enough to help us recognize the situation in which we currently reside. Simply put, you can have great solutions, but if they are applied in the incorrect context, they will be worthless or worse, harmful [11].

4. SUMMARY

The Cynefin framework can be used to identify the best suited software development methodology and practices for each of the identified situation. Executing best practices and expert models in a disciplined fashion makes sense when dealing with Simple and Complicated domain issues. Issues in or bordering the Chaotic domain requires fast action to prevent a catastrophe from happening. Complex domain issues require experimentation as a way to learn about and understand its patterns, patience is required to let the system find a practical and acceptable solution.

Software development is a rich domain, containing many aspects, a large percentage of which can be classified as complex. The interaction between these aspects is also complex. Just as we have benefited from treating software development as complex, and taking advantage of the toolbox of social complexity, namely the Cynefin framework, so the field would benefit from a multi-ontological approach, taking the best techniques for the various domains, and combining them in an appropriate and flexible manner. More work is needed to reach a deeper understanding of the inter-workings of agility and complexity within a Cynefin context [10].

5. REFERENCES

[1] Kurtz, C. F. and Snowden, D. J.. 2003. The new dynamics of strategy: Sense-making in a complex and complicated world. IBM Syst. J. 42, 3 (July 2003), 462-483.

[2] Boehm, B.and Turner, R.. 2003. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[3] Coleman G. and O'Connor, R.. 2008. Investigating software process in practice: A grounded theory perspective. J. Syst. Softw. 81, 5 (May 2008), 772-784.

[4] Dettmer, H. W., 2011. Systems thinking and the cynefin framework—A strategic approach to managing complex systems, Port Angeles, WA: Goal Systems International

[5] Snowden, D. J., & Boone, M. E. 2007. A leaders framework for decision making. Harvard Business Review, 69-76.

[6] Lepmets, M., McBride, T. and Ras, E.. 2012. Goal alignment in process improvement. J. Syst. Softw. 85, 6. 1440-1452. DOI=10.1016/j.jss.2012.01.038.

[7] Salo, O. and Abrahamsson, P. 2007. An Iterative Improvement Process for Agile Software Development, Software Process: Improvement and Practice*,* vol. 12, pp. 81-100.

[8] Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., and Abrahamsson, P. 2014. Software development in startup companies: A systematic mapping study. Inf. Softw. Technol. 56, 10, 1200-1218. DOI=10.1016/j.infsof.2014.04.014

[9] Thorpe, R. Holt, R., Macpherson, A. and Pittaway, L. 2005. Using knowledge within small and medium-sized firms: a systematic review of the evidence, Int. J. Manage. Rev., 7 (4), pp. 257–281

[10] Pelrine, J. 2011, On understanding software agility – a social complexity point of view, E:CO,Vol. 13 Nos 1-2, pp. 26-37.

[11] Hobcraft, P. 2015. The Use of the Cynefin Model for Innovation, Paul4innovating's Innovation Views, Blog 19 June 2014, available at http://paul4innovating.com/ [Access March 2015]

[12] Kim, J., 2015. How Complexity Domain Impacts Software Development Process, Mobile Game Industry Points of View, Blog 9 March 2014, available at http://quarterview.com/ [Access March 2015]

[13] Clarke, P. and O'Connor, R.V. 2012. The situational factors that affect the software development process: Towards a comprehensive reference framework, *Information and Software Technology,* vol. 54, pp. 433-447.

[14] Samios, H. 2015. Software development, Scrum and Cynefin, Blog 3 December 2013, available at http://www.hanssamios.com/ [Access March 2015]

[15] Lepmets, M., O'Connor, R., Cater-Steel, A., Mesquida, A.L., and McBride, T. 2014. A Cynefin based approach to process model tailoring and goal alignment. in Proceedings of the 9th International Conference on the Quality of Information and Communications Technology. IEEE Computer Society.