Resilience Implementation Structure for Complex IT Systems[1]

Arvind Mundra

The expanse of domains, where the principle of project management can be applied, almost overlaps entire work space in the modern world. When someone refers to project management in typical IT projects, most of us presume that the project would be about developing delivering new exciting features to delight the customer-base. From time to time, we come across projects where technology infrastructure is the core focus. The specific goals of such projects may vary but they generally fall into the broader categories of improving the efficiency of infrastructure and alignment for a new solution architecture. Then comes third type of projects where the goals are much more nuanced and contextual (though not less critical) like security, reliability, resilience etc.

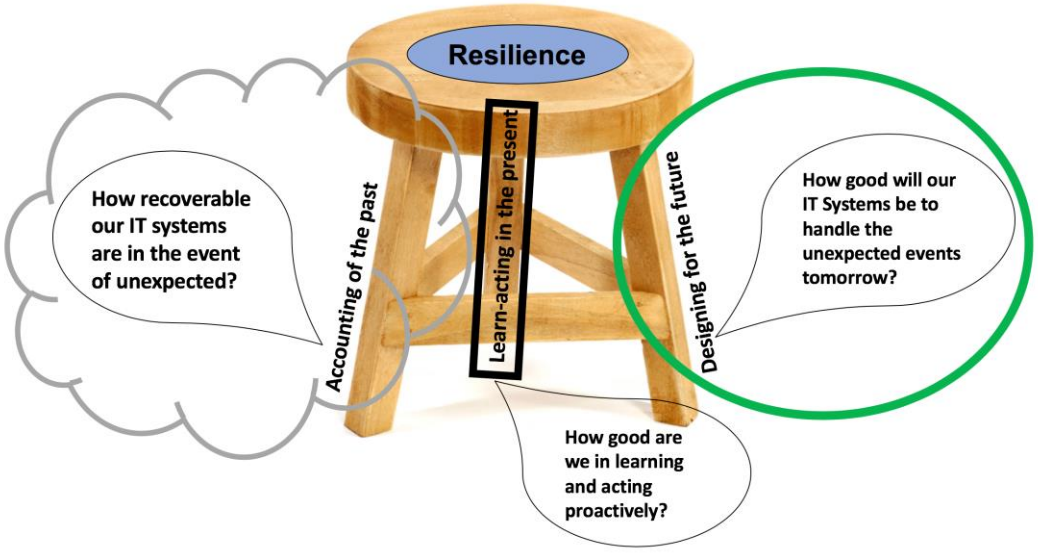
These projects are created to strengthen a particular area of operating complex IT systems so that value delivered by typical software development and infrastructure projects reach the customer-base with minimal risks and challenges. Resilience Implementation is one such area for project management, considering the new world of impatient customers with social media to take down any company's hard-earned reputation in minutes.

Improving Resilience has been key focus of most (if not all) large organizations with complex IT systems. Transforming the tenets of Resilience science into value-adding Resilience Engineering requires a comprehensive and adaptive structure. We need comprehensive structure because without considering all parts of systems and all dimensions (like past, present and future; and IT, human and organization; and strategical, tactical and operational etc.), it will be difficult to achieve meaningful resilience. Also, only an adaptive resilience framework could sustain itself in ever-changing technological and organizational landscape.

The proposed resilience implementation structure is meant to be one of many ways an organization can structure its focus and efforts to embark upon a resilience journey.

[1] How to cite this article: Mundra, A. (2019). Resilience Implementation Structure for Complex IT Systems, *PM World Journal*, Vol. VIII, Issue I (January).

Triangle of Resilience


Resilience
How recoverable our IT systems are in the event of unexpected?
Accounting of the past
Learn-acting in the present
Designing for the future
How good will our IT Systems be to handle the unexpected events tomorrow?
How good are we in learning and acting proactively?

- **Accounting of the past**

Past can sometimes be cruel – It is true of human story and so is of IT systems. Technical debt is broader often-used term which encompasses not only resilience challenges (both unintentional and intentional causes like not understanding system level behaviors and crunching the time-to-market etc.) but also security, operability, scaling and other categories.

The fundamental question that we ask of IT system which is given to us as heritage:

How recoverable our IT systems are in the event of unexpected?

Now it can be difficult to measure as many of IT systems comes with stack of legacy infrastructure which do not lend themselves very well for the purpose of measurement or have capability to allow probes to monitor them (or in some cases, this monitoring itself impacts measurements as it becomes extra process to manage for the system).

Also, it is challenging to decide upon what measurements should we choose and how to add meaning to this measurement quantification. If we take myriad groups/teams that are supporting a large/complex IT system into consideration, a common understanding and association to these quantification measures does not remain a nice-to-have but becomes a must-have.

It is only rationale that this stick of measurement remains simple - not only that it can be applied to different components, sub-systems and systems with equal effectiveness but also it should be easy to keep everyone aligned in terms of understanding. The measurement of resilience of an in-the-hand-of-customer-system needs to be free of external measures like customer traffic, security threats, number of features, in-house development or vendor integration, new or old technology stacks, age of the product/service and nature of hacking attack etc. One or more of these parameters may certainly influence the resilience of the system from one week to next (and from one month to another) but cannot define it.

In a broader context of a complex IT system, following can be a good working definition of resilience –

> ***What is the strength of system (with all its parts) to recover from unexpected disruptions with typically no external input?***

Now how do we quantify and measure it? One simple way is to count all potentially customer-impacting downtimes as a number and then record how many of those got resolved with planned response (either automated or human intervention). That ratio is a simple measure of how resilient the production IT system is.

$$\text{Resilience} = \frac{\text{\# of potentially customer-impacting downtime events with planned resolution response}}{\text{Total \# of customer-impacting downtime events}}$$

To find all potentially customer-impacting downtimes events, we need a suite of test cases which are expressly designed to test resilience of various sub-systems, components and integration points. This can be done by system architects/engineers with contribution from tech leads of different sub-systems and components. It should be supplemented by actual customer-impacting downtime events reported by operations. In complex IT system, just like most other things – this is easier said than done. The leadership drive and resource allocation are the key to success on this.

This formula is not definitive and applicable for all IT systems scenarios but just a guideline. It can be tweaked to best serve in specific context of an organization and its IT system but once a definition (and method of its quantification) of resilience is established, it should be tracked over time. The gaps of definition and inefficiencies of measurement will get discounted in this trending as the data will be more or less similarly compromised. In case, where the data points are found to be disparate to greater degree, the definition of resilience and methods of measurements need to be re-evaluated. If needed, the events of customer impacting downtime can be further tightened with certain number of impacted customers per event e.g. only downtime events where more than 1000 customers were impacted would be counted for this formula. This can be a good start and as the organization gets better in addressing big-ticket items of resilience, which shows up in improving trend of resilience score, this criterion can be further tightened to catch smaller improvement opportunities.

It needs to be stressed that the goal of this exercise in not to reach a number but to evaluate those underlying data points to identify the areas for focus and ways of boosting resilience. The results need to be analyzed by **system architects or engineers** with broader understanding of system and result should lead to identification of sub-systems/components that need resilience boost and how best to achieve that (which connect this leg to next leg of **learn-acting in present**).

Also, this analysis will provide info on how the organization need to change in software development practices so that future code addition to the production system will not have similar deficiencies (which connects this leg to third leg of **designing future with resilience**).

- **Learn-acting in present**

They say, present is all we got. Not quite. Present is quite ephemeral. Only way, it can be captured is through action and this is what this leg of resilience is all about. Developing insights from the learning highlighted in previous leg, acting upon them to plug the holes of existing system and continuous measuring are key features of this leg.

The definitions and measurement quantification are taken care of in the previous leg, so there is no new formula for this leg. As long as we keep measuring frequently and learn-act from the trend, we are on the path of Kaizen.

The methodical analysis of data lying behind the trend should guide us to key areas which would benefit highest from resilience boost perspective. The **DevOps lead** in the organization should oversee this leg, as they understand the ground reality of existing systems best and have first-hand context of what would alleviate most pain with least efforts with focus. While working in tandem with architects or system engineers (who are the lead of previous leg), these DevOps leads would unravel of mysteries behind the resilience trends and translate them into short burst of devops efforts which will not only "stop the bleeding" but also chip away at certain (may not be all) resilience-related inefficiencies. Now these short bursts of devops efforts may not take care of all systematic and/or architectural inefficiencies (which will be taken care by the third and last leg), though they can certainly put efficient work-arounds and short-term fixes in place. The knowledge and awareness generated out of this leg will act in-part as foundation for next leg.

Ongoing reporting of resilience (score) is key to keep the focus on. Without keeping scores, you cannot have well-played game and same is true for the resilience. An automated test suite of resilience test case can have additional functionality of automatically generating a resilience score (on a weekly or monthly basis) and disseminate that info to all.

- **Designing future with resilience**

Future is exciting. Mostly. This is where we can design the past and present of tomorrow.

Things that past taught us while we were in present can equip us the power of knowledge and intuition of intelligence so that we can design our future. The work on this leg is not as much heavy-lifting as other two legs. The **leads from DevOps/Dev** team can provide oversight to this leg's effort. There should be training program to instill the importance of resilience to all people working on future products/services. The resilience should be part of coding (and peer code review) which should be tested from unit test level onwards. The functional and service level testing both should include resilience as a category of tests, just like security or operability.

Just like quality and security, we have to build in the resilience. If that means, curbing our age-old tendency to cut corners on lines of codes and/or hardware spend, be it so.

Instituting the culture that resilience cannot be an after-thought is key to build the system for tomorrow with reduced probability of hurting business, dragging the customer confidence down and damaging company's reputation.

Each new release (before it goes to trial/production) should have a resilience score associated with it (based on execution of suite of test cases focused on resilience) and go/no-go decision should consider resilience score as part of decision-making criteria with simple question of – "*will this new release add to debt of resilience of overall system or will it improve it?"*

These three legs of the resilience framework can provide an implementation structure to start on often over-whelming journey of measuring, improving and building resilience in complex IT systems.

About the Author

Arvind Mundra

Philadelphia, PA, USA

Arvind Mundra has been working Agile Project Management for last 20 years in various roles. After spending 6 years as C++ developer, Arvind became interested in aligning project management to the maximum value creation from a software development team's work. For last many years, Arvind has focused on training software development teams on Agile Project Management and coaching them to build the self-sustaining discipline.

Arvind has gone through multiple certifications from both PMI and Scrum Alliance and has contributed to the pilot program of PMI-ACP certification and has been active in local Agile user groups.

Arvind can be contacted at arvind.mundra@yahoo.com