Resilience Engineering as an IT Cultural Discipline

To enable always-on and always-available digital business, IT organizations must become more operationally resilient to enhance system and application stability, which fuels service reliability and boosts reputational integrity.

Executive Summary

Today’s customers expect information to be readily available irrespective of the channel and time of day when they require access to it. To deliver on this demand, leading digital businesses depend on always-on and always-available IT systems. Traditional IT strategies have proven to be ill suited and expensive to meeting this requirement. The culprits: the ever-expanding landscape of systems, platforms and services, as well as the related application architectural changes required to keep a digital business humming at peak performance.

Resiliency engineering is an effective and much needed approach to ensuring system and application reliability. This white paper examines the increasing relevance of resilience in an enterprise IT landscape, how it supplements standard availability management practices, and how its application can enable enterprise IT to build and adopt a culture of resiliency.
THE EMERGING DIGITAL BUSINESS CHALLENGES & RISKS LANDSCAPE

The business challenge and risk landscape has radically changed with the emergence of microservices, cloud and server-less computing, which power today’s highly distributed enterprise. While applications and architectures have become more distributed and connected, customer expectations on downtime tolerance is approaching zero.

Key business challenges that this shift creates include:

- **Maintaining global business reach.** Multinational organizations must address the needs of customers in different geographies, many of whom have varied demands and interests. Moreover, they must also do this across time zones, essentially keeping their systems running continuously and providing the same level of service to all customers.

- **New channels that break availability barriers.** Channels are the key touchpoints through which an organization liaises with its customers. The increase in digital channels use (i.e., mobile, tablets and laptops) enables customers to access the desired products and services anywhere and at any time they want. Add to this connected products, intelligent appliances and voice assistants, which have expanded the number of channels through which an organization’s services or products can be accessed. Consider a connected fridge that can detect a household running out of milk and can automatically place an order with a preferred store, and the possibilities are endless.

- **Uptime = revenue.** At many companies, IT has shifted from being a business enabler to being the business itself. Examples such as Amazon.com, branchless digital banks such as Ally, Root car insurance company, etc., have transformed the way traditional business services are consumed. Given business’s ever-increasing focus on digital, the downtime of one organization is an opportunity for another.
For example, if a consumer orders pizza for dinner and finds a particular site is not available, she will immediately switch to another pizzeria. Black Friday shopping historically provide examples of how high transactional volumes can cause downtimes for unprepared retailers. Conversely, highly available and responsive systems can have a positive impact on brand equity and increase customer retention. This can result in a virtuous cycle: a boost to income and profit means more capital available for investment.

- **Bad news spreads like wildfire.** With the advent of Facebook, Twitter, WhatsApp, etc., any news on business impact spreads quickly to all channels. For example, shoppers were aware of Amazon’s website crash on its Prime Day Sale 2018 within minutes of its occurrence.

- **Regulatory implications and penalties.** Due to the increasing number of regulations and need for operational transparency, organizations are increasingly adopting consolidated and harmonized sets of compliance controls. Organizations not compliant or not adhering to policies established by the local government could end up paying huge penalties or get entangled in unproductive lawsuits.

- **New technologies that redefine boundaries.** Innovation drives differentiation. This gives a disproportionate advantage to the use of disruptive technologies. Many organizations are on a constant lookout for solutions that offer top- and bottom-line growth. These come in all shapes and sizes – providing better customer experiences, leveraging data to provide better insights and so on. Efforts are underway in the areas of automation, Internet of Things (IoT) and artificial intelligence (AI) that can improve the speed and accuracy of service and reduce human intervention.

- **Reliability is a new differentiator.** Reliability measures availability, accuracy, and delivery of a service within a time threshold (i.e., responsiveness). This defines how consistent organizations are in delivering services to customers. The service could be a website, business process, an application, or a series of processes.
Similarly, the risk landscape is also undergoing a fundamental shift:

- **Dynamic business demands.** It is imperative that applications and the supporting infrastructure are dynamic and elastic enough to respond to changing business needs and in many cases can predict and prepare themselves for the changing business needs. For example, the expected load on Black Friday increases annually.

- **Infrastructure failures.** The highly distributed cloud-native applications will only increase the probability and impact of systems’ failure. It is therefore essential to ensure that applications can tolerate failures caused by cloud providers. A recent example was the impact of a Google cloud failure on services provided by companies such as Spotify and Snapchat.

- **Application faults.** Several lines of code must be executed for a single business functionality. A poorly designed or coded application that is not well tested could break under certain business conditions.

- **Human errors.** Microsoft offered a human error explanation for a major outage of Azure cloud services on November 18, 2014. The downtime left some customers of Azure Storage and other offerings without Internet service. Microsoft’s explanation for the outage makes clear that its own procedures were not followed. Poorly documented or standard operating procedures that are not followed can lead to downtime, and must be avoided at all costs.

- **Malicious attacks, natural disasters, external factors like legal, political, etc.**

- **The myth of the cloud.** With cloud, all your problems related to resilience somehow magically vanish. For example, auto scaling and auto-restart of virtual machines (VMs) prevent downtime by conducting proactive monitoring and predictive analytics to detect load vs. available resources. While the cloud does help to some extent, fault awareness and fault tolerance should be designed and built into applications rather than continuing to depend on infrastructure for an after-the-fact resolution.

**RESILIENCY ENGINEERING: A PRIMER**

As the digital business economy operates 24x7, so must the indispensable IT systems that power it. Existing philosophies for achieving this are to eradicate all possible failures by building robust systems, which is almost impossible due to complex legacy architectures, the demand for accelerated time to market, and the aforementioned increasing risk landscape, etc.

Given these shortcomings, we believe that IT organizations must embrace failure and enable faster automated recovery by identifying and addressing all types of failures during the development lifecycle and engineer recovery mechanisms to ensure continuous service.
organizations must embrace failure and enable faster automated recovery by identifying and addressing all types of failures during the development lifecycle and engineer recovery mechanisms to ensure continuous service. This is how we think organizations should build highly resilient systems for the modern digital age.

We call this construct “Resilience Engineering.” This means taking a systematic approach to software development aimed at understanding all failure points within an application/enterprise and the supporting infrastructure, and addressing every failure with both a tolerance and recovery strategy. This results in applications that are fault-aware, fault-tolerant and self-healing.

Why Engineering Resilience is Critical

Since multi-tier modern digital systems consist of several applications built with tools such as AngularJS, jQuery, ReactJS, Spring, Hibernate, .NET Core, etc., their vulnerabilities or availability issues often occur between tiers. For example, in an e-tail application that is accessed by a multiplicity of devices — mobile phones, laptop/desktops and in-store — a transaction must traverse multiple layer content servers, service layer connectors, a database tier, mainframe systems of record and payment gateway. So latency or a fault that arises in any of the layers will break the chain to complete the business transactions, which directly affects operational performance (i.e., no order, no revenue, means no business).

Resiliency has therefore become of critical importance as organizations implement software that runs across multi-devices, multi-tiers and multiple technology infrastructures to provide continuous service to customers. As a result, IT organizations must adopt resilience engineering processes in their software design cycles and embrace a culture of continuous improvement. There are multiple ways to enable an application to return to its original state after a failure.

Key principles to make an application more resilient include:

- **Defining alternate business paths.** Business processes must provide for alternate paths to follow in the event of faults. Such alternate fault paths should be defined for all application components (internal and external) that form part of the business process execution. For example, for an online retail store locator, the process can have multiple options such as Google Maps, MapQuest, etc., to share geographic location of a store. This provides a fallback option for system designers to direct the user to Google Maps when MapQuest is unavailable.

- **Enabling fault tolerance.** A failure in a dependency system should not break the end-user experience. For example, in an online banking system, a mainframe database is a dependency system that holds all account transactions for customers. Searching for a payment transaction on a specific statement period should not be interrupted by mainframe database unavailability.
• **Reaction to failures.** Applications should take corrective action when a dependency fails. In the same online banking example, the website should show a meaningful error message to the customer who is looking for a statement during a mainframe database outage. It should then email the statement once the database is available.

• **Tracking the status.** The system should have the capability to identify the failed component, the number of customers impacted, and report when it became available for the root-cause analysis and deciding corrective long-term solution to avoid future occurrences.

**ENABLING A RESILIENT IT ENGINEERING CULTURE**

IT organizations need to consider enabling resilience engineering in their process and culture. This is especially true when there is an increase in the number of priority 1 or 2 incidents; these involve outages, availability impacts, slowness, etc., in production, or businesses going through a transformation such as mergers or acquisitions or cloud migrations, or when service-level agreements (SLA) or operation-level agreements (OLA) are being redefined. To create a resilient IT engineering culture, the organization must assess its current state of maturity in application and infrastructure resilience, socialize the concept with key teams, build knowledge, and invite its members to actively participate and maintain it.

The following are important characteristics of resilient IT organizations.

• **Responding to failure situations.** IT organizations must adapt their processes and procedures to proactively respond to failure conditions. This requires a runbook (a compilation of routine procedures and operations that the system administrator or operator carries out) to act on failure conditions, with possible steps to recover the applications from specific failures.

• **Monitoring the environment.** The IT organization should instrument its environment with the right set of monitoring and alerting application performance management (APM) tools, such as Dynatrace, AppDynamics, etc., to monitor both internal operations as well as external interfaces for potential failures.

• **Anticipating failures and improvement opportunities.** An IT organization should not only focus on resolving current impacts, but should also anticipate possible future occurrences that may affect the regular business
and operations stability. If an outage occurs to an e-commerce site payment gateway during peak seasonal load (e.g., Black Friday) due to high volume of access requests, then the team must assess the feasibility of adding another payment gateway as a fallback option to provide continuous service.

- **Learning from experience.** The IT organization’s resilience can be improved by learning from past experiences. It’s particularly important to learn from successful responses to adverse events such as fail fast and recover automatically to internal/external system failures and to enable swift service restoration.

Figure 1 depicts the steps for bringing cultural change to an enterprise IT organization.

**Enabling Cultural Change**

Deeply embedded processes cannot be replaced with simple upgrades, or even with major overhaul efforts. Before bringing in a new process to the culture, it’s important to assess the organization’s current resilience maturity level by studying the various attributes of business process, people, technology and operations and rate them against industry-standard maturity models.

Once the current state of resilience maturity is understood, tailor the roadmap with specific actionable steps to raise levels to mature the organization’s resilience. Identify the relevant talents such as resiliency architects and resilience engineers to lead the actionable items from the resilience maturity roadmap to achieve common enterprise resilience goals.

Implementing resiliency engineering requires IT organizations to add new actions to their software development lifecycle. These include:

- **Resilience assessment:**
  - Assess current resilience maturity.
  - Identify gaps and rank based on criticality.
  - Arrive at roadmap to improve resilience maturity.

- **Resilience engineering:**
  - Introduce resilience constructs in application architecture and design.
  - Design and implement self-CHOP\(^3\) techniques integrating across applications, containers and monitoring solutions.
• Resilience validation:
  » Validate resilience by introducing various failure scenarios (faults, load and others) and observe application behavior.
  » Identify areas of improvement and provide feedback to engineering.

Socialize & Operationalize Resilience Engineering

IT organizations must help architects, developers, testers, infrastructure and operations teams to understand, commit to, accept and embrace resilience engineering. It is a commonly held view that behavioral change follows mental shifts. The following important factors can help advance adoption, acceptance and successful implementation of resilience engineering.

• Create documentation on the process changes with additional activities and ownership in existing development and maintenance cycles, resilience design patterns and implementation guidelines, validation tools and business processes.
• Communicate the change in processes regularly and early.
• Provide opportunities for employees to better understand the impact and for groups to get a feel for the different way of doing things. For example, implement an in-memory caching as a fallback for frequent database calls.
• Allow for suggestions/feedback.

Learning & Development

Trying to change a culture purely through top-down messaging, training and development programs and identifiable cues seldom alters people’s beliefs or behaviors. Build competencies by inviting resilience architects or consultants to share their experiences or best practices and encourage people to learn resilience engineering tools, designs, validation platforms and more. IT organizations should consider creating task forces for communicating shared goals to the team through pilot projects that add resilience engineering to their project lifecycles.

Integrate: Building Resilience into IT Systems

Building IT systems for resilience means creating requirement definitions, which must be tracked/enforced throughout the project, at the beginning of the software development lifecycle. IT teams should define, design and program for faults, validate the faults in IT systems by creating faults, let the system fail, and make sure it’s resilient during fault conditions.

The fault identification and alternate business path to provide continuous service should start from defining business functions and processes, designing to handle the faults followed by coding for failures, and then validating failure scenarios and system responsiveness to faults.

Measure Resilience

Understanding the attributes and dimensions of resilience provides guidance to measure the adoption and effectiveness of resilience engineering implementation. As illustrated in Figure 3, it is
recommended to shift focus from maximizing mean time to failure (MTTF) to minimizing mean time to recover (MTTR) to build highly resilient systems.

Resilience can be quantified as time at which a system returns to functional mode after a disturbance from normal operations. For example, Resilience = f (Robustness, Rapidity)

- **Robustness** is the measure of impact to system function in fault mode.
- **Rapidity** is the time taken to recover to normal mode of operation (time to discover, time to isolate, time to fix and time to recovery).

The system function, or quality of a software system, is defined by performance and functional availability. Functional availability covers accuracy of all functional and data components that could be measured at every business outcome level. Criticality, volume of users accessing the functionality, end-to-end completeness and performance are critical factors in measuring the availability.

Resilience improvement can also be measured from increase in MTTF and reduction in below KPI metrics over time:

- Mean time to discover (MTTD) and MTTR.
- Response time.

**SUSTAINING SYSTEMS THROUGH CONTINUOUS FEEDBACK**

Building a highly resilient system is a journey; at each stage resilience is incrementally built into the system. The process of incrementally building highly resilient systems involves continuous resilience engineering, periodic resilience audits and changes to the application systems. We suggest a revamp to the resilience engineering and validation process from the feedback of internal and external customers using the live systems. This can help elevate the maturity of IT resilience in the organization.

The motivation for resilience engineering stems from the organization’s need for more stable and reliable IT operations. As with other disciplines, a point of diminishing returns will be reached when the investments in resilience can no longer be justified. An organization should continue to evaluate the risks on an ongoing basis and strike the right balance between the budget and engineering efforts involved in managing resilience. The following factors influence the need for a resilience strategy and the management elements therein:

- Major change in business drivers and volumes.
- Application architecture and technology revamp.
- A hardware refresh.
- A data center expansion.
- Public cloud migration. Service subscription from cloud-based vendors (e.g., Salesforce.com, etc.).

**LOOKING AHEAD**

Despite ever-accelerating technology advances over the past decade, achieving application stability is still fraught with complexity and prohibitively high costs. Resilient applications provide much more than reliability: they help to build a continuous innovation process by predicting the failure paths and design for failure to move fast and stay ahead of availability goals to build always-available systems and provide continuous service to customers.

To get there, IT organizations must assess the current cultural situation, and change behavior and mindsets to enable their business, process, tools and technology to deliver enhanced quality of service to users inside and outside the enterprise.
QUICK TAKE

Resilience Engineering at a Large North American Cable TV Outfit

This client faced stability issues in its critical applications, and the cost of addressing downtime had increased 22% year over year, impacting profitability. The company sought to infuse resilience engineering and resilience validation practices into its release cycle to reduce application downtime and improve stability.

We assessed the company’s digital IT landscape and provided solutions to improve performance and resilience such as bringing process and cultural change to include risk-based evaluation of production releases, performance-driven development, and resilience engineering and validation. The goal was to achieve zero downtime and fault-tolerant customer-facing applications in the digital portfolio.

Our Approach

Our resilience engineering team worked with the customer teams to drive fault-vulnerability analysis to identify fault vulnerabilities in each application and engineer tolerance and recovery capabilities before they were released to production. This was accomplished through implementing resilience design patterns in the applications and embedding a resilience engineering methodology into their Agile/DevOps-based development pipeline.

Customer Benefits

Post implementation of resilience features in their IT organization, the following benefits were measured:

- 20% reduction in number of non-functional incidents year to date (YTD).
- 32% reduction in incident duration (MTTR), YTD.

Accomplishments

During the course of the engagement, the team built many automation frameworks and solution accelerators. Key achievements included:

- Created two latency bots to simulate high latency conditions and five chaos bots to simulate failure conditions in non-production environments.
- Implemented resilience engineering and validation in ten business-critical applications.
- Identified approximately 225 non-functional defects on or before the QA cycle and zero defect leakage to production.
ENDNOTES

3 Self-CHOP (Self-Configuration, Self-Healing, Self-Optimization and Self-Protecting)

REFERENCES


ABOUT THE AUTHOR

Premkumar Balasubramanian
Vice President, Cognizant Digital Engineering

Premkumar Balasubramanian is an Assistant Vice President within Cognizant Digital Engineering, where he heads the Performance and Resilience Engineering practice in North America. He has over 19 years of experience in performance and resilience engineering and has led large programs for many customers spanning across different business units. He has been an active speaker in STAREAST and STARWEST conferences and at many customer and Cognizant events in the field of performance engineering. He can be reached at prem@cognizant.com | www.linkedin.com/in/prem-balasubramanian.

Sivakumar Kalyanaraman
Senior Technology Director, Cognizant Digital Engineering Performance Architecture Consulting and Engineering (PACE) Practice

Sivakumar Kalyanaraman is a Cognizant Senior Technology Director and heads resilience engineering services within the company’s Digital Engineering Performance Architecture Consulting and Engineering (PACE) Practice. He has over 18 years of experience in performance/scalability/availability/resilience engineering and tuning of large IT systems in Java/C on UNIX and Windows systems. He heads the resilience engineering services and works with various customers in assessing, designing and implementing resilient IT systems. He can be reached at Sivakumar.Kalyanaraman@cognizant.com | https://www.linkedin.com/in/sivakumar-kalyanaraman-b31b2142.

Sivasubramanian Bagavathiappan
Senior Performance and Resilience Architect, Cognizant PACE Practice

Sivaubramanian Bagavathiappan is a Senior Performance and Resilience Architect within Cognizant’s PACE Practice. He has over 12 years of experience in software development, testing, performance, scalability, capacity and resilience management of large enterprise applications in various UNIX and Windows environments across multiple domains. Sivasubramanian specializes in performance engineering strategies and transformation to performance-driven development lifecycles. He can be reached at Sivasubramanian.Bagavathiappan@cognizant.com | https://www.linkedin.com/in/sivasubramanian-bagavathiappan-155738102.
ABOUT COGNIZANT
Cognizant (Nasdaq-100: CTSH) is one of the world's leading professional services companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 195 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.