**AGILE PARALLEL DEVELOPMENT EBOOK SERIES:** Chapter 2

# Five Dev/Test Constraints Impacting the Speed, Cost and Quality of Your Applications

# New Pressures, More Constraints

It's no secret that in the Application Economy, leading organizations are creating competitive advantage by developing new digital innovations and combining them with existing goods and services.

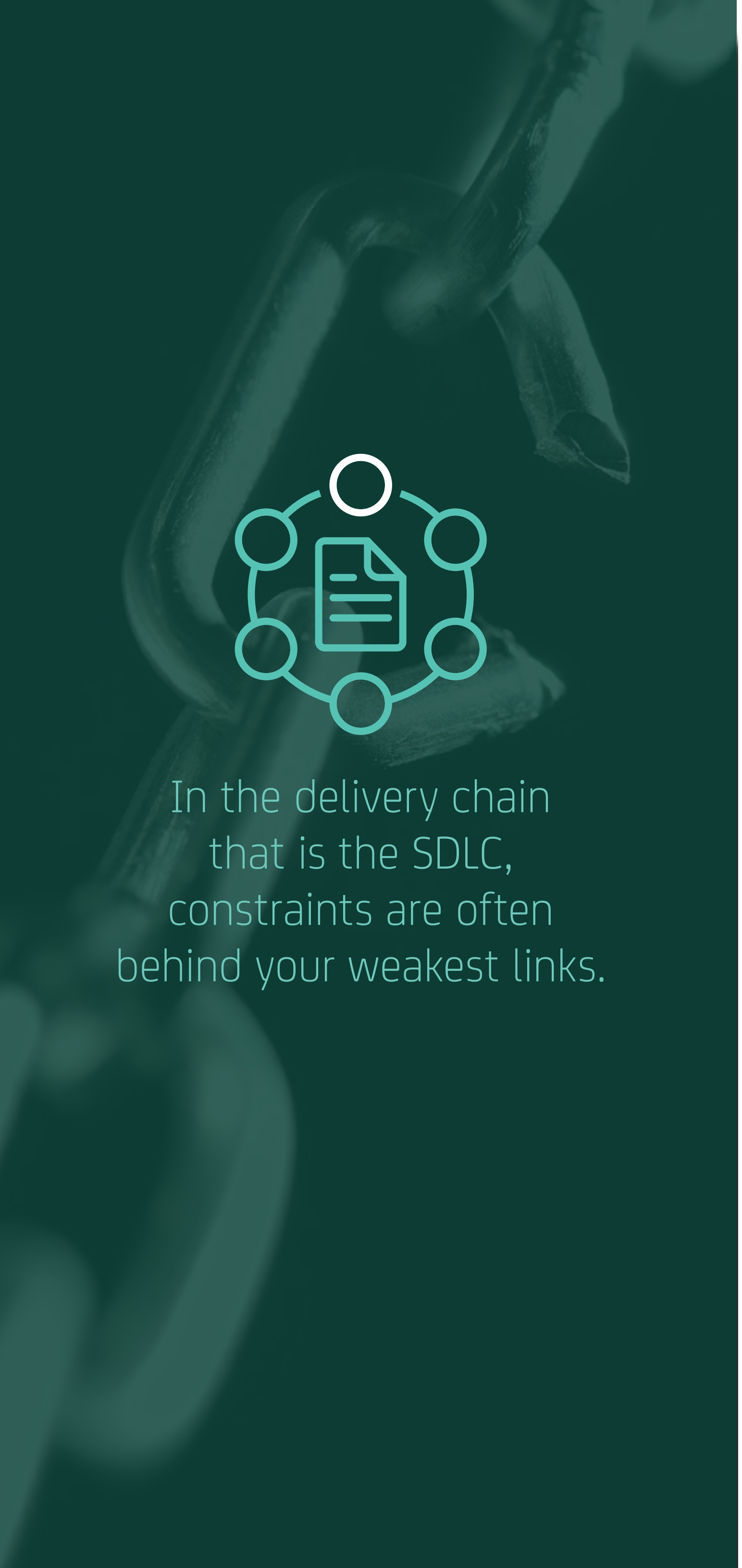
Known as **Digital Transformation**, this initiative has caught fire with companies hoping to meet the ever-increasing demands of users who have been trained by the Internet, mobile devices and social media to expect fast, flawless performance from every digital service they consume.

As part of this transformation, IT organizations are being pressured to deliver new applications and services at an accelerated rate, while keeping software quality high and dev/test costs to a reasonable level. Despite their best intentions, however, dev and test teams often struggle to meet one or more of their speed, cost and quality goals—and they have the persistent challenge of constraints to thank for that.

## WHAT IS **DIGITAL TRANSFORMATION** ?

*Altimeter* defines it as "the re-alignment of, or new investment in, technology and business models to more effectively engage digital customers at every touchpoint in the customer experience lifecycle."[1]

[1] Altimeter Group Digital Transformation Survey, 2014, N=59.



In the delivery chain that is the SDLC, constraints are often behind your weakest links.

# How Constraints Limit Development Potential

You've probably heard the idiom that a chain is only as strong as its weakest link. For dev and test teams, there is a similar relationship between the software delivery lifecycle (SDLC) and constraints.

Coined by Eliyahu M. Goldratt in his book, *The Goal*, the theory of constraints states that **within any manageable system, there are constraints that can impact the achievement of a particular goal**. For example, dev and test teams encounter numerous environment, data, testing and visibility constraints within the individual stages of the SDLC—each of which creates specific delays or complications that prevent the delivery of high-quality services at expected costs and speeds.

This means that even if the business and users want better apps delivered faster, IT will only be able to develop and test as quickly and frequently as the constraints within the SDLC allow. As such, if you expect to reach new speed, cost and quality levels, you must understand the common constraints that plague the SDLC—and how to minimize or potentially eliminate them.

# Five Common Constraints Across the SDLC

**Identifying and weeding out constraints across the SDLC can often seem like a game of "whack a mole"—when you knock one out here, others pop up there, and they can be difficult to predict. Here are some common types of constraints to look out for when reviewing your development and testing practices:**

1

## Composite Application Constraints

Modern applications are assembled from numerous APIs, UIs, components and services—many of which you may have limited access to for development and testing activities. And when these components reside with a third party, you often have to pay access fees that can quickly add up over time.

2

## Environment Constraints

You have multiple application delivery teams but only so many dev and test environments on which they can work. As a result, teams inevitably experience unproductive idle time as they wait for needed systems or environments to become available.

# Five Common Constraints Across the SDLC

3

## DATA CONSTRAINTS

In order to ensure that an application will perform up to standards in production, you want to test it with data that is as close to “live-like” conditions as possible and has the breadth to cover all use cases. And if test teams don’t have easy access to such data, they have to manually create production-grade test data sets, which adds additional costs and time to projects.

4

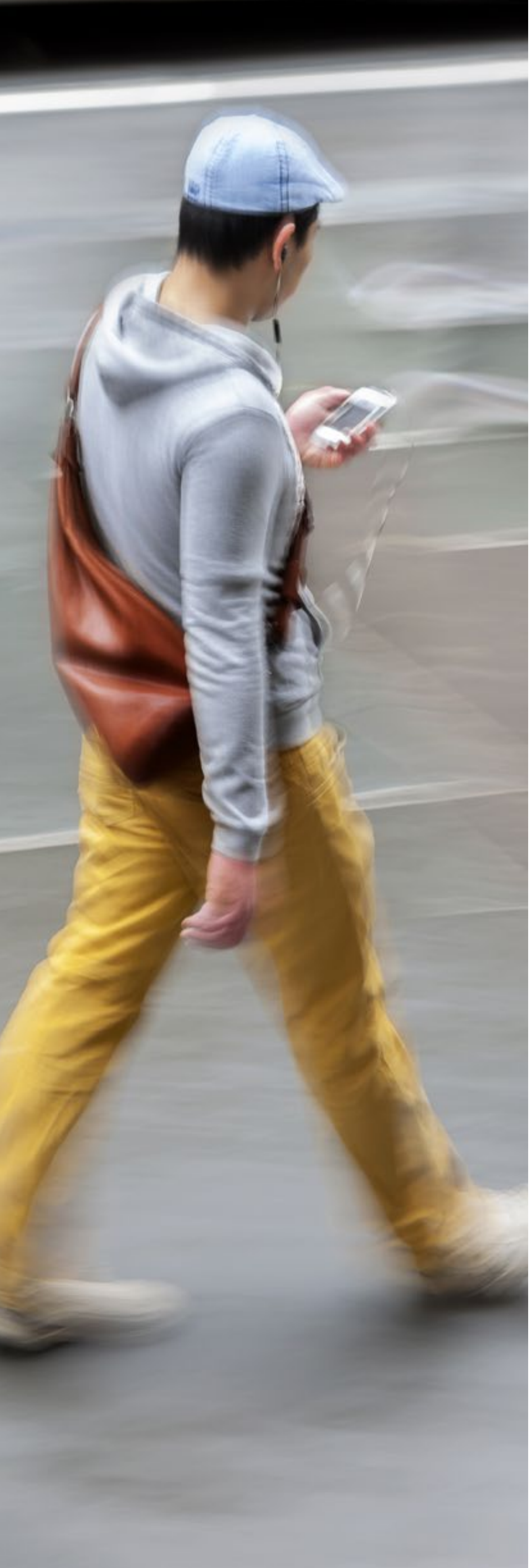
## TESTING CONSTRAINTS

Many IT teams believe their testing procedures are automated, but it’s only the testing itself that proceeds automatically. In order to get to that step, they’re still having to manually develop scripts and code to drive the automation—and these test assets can rarely be shared across test stages (e.g., unit, functional, regression, integration, performance, etc.), creating a need for duplicate effort.

5

## VISIBILITY CONSTRAINTS

When testing reveals defects in an application, IT teams must pinpoint the root cause of the issue before they can fix it. In a composite application environment, this is easier said than done—especially if they lack visibility into the end-to-end transaction path and struggle to unearth actionable insight.



# How Dev/Test Constraints Impact Your Business

**While constraints manifest as technical challenges that IT teams must address within and around the SDLC, they also impact the business as a whole in a few significant ways.**

### SLOWED INNOVATION

Coming up with an innovative software idea is only half of the equation; the other is getting it to market quickly and at a quality level that will create competitive advantage and exceed users' expectations. However, many of the previously discussed constraints contribute to idle time and manual effort that extend project timelines—effectively putting the brakes on innovation and giving competitors the chance to beat you to market.

### REDUCED QUALITY

When teams don't have timely access to live-like test assets, they cannot test as efficiently and comprehensively as is necessary to ensure production-ready performance. And when low-quality applications hit the market, they turn off customers, risk damaging your brand and require significant dev/test rework.

### INCREASED COSTS

Having to work around constraints creates inefficiencies in the SDLC that lead to hard and soft costs for the business. For example, when coders use their high-value time to manually create low-value scripts or mocks, the "human cost" of delivering an application goes up. There's also the present known and future unknown revenue impact that a bug-ridden service has when it's prematurely released to the market.

# Solving the Constraint Problem with **Agile Parallel Development**

**In order to limit the effect constraints have on software development speed, quality and costs, leading organizations are adopting development tools and methodologies that are both agile and parallel.**

## With Agile Parallel Development, you can reduce or eliminate:

**COMPOSITE APPLICATION AND ENVIRONMENT CONSTRAINTS** by **modeling and simulating the behavior and performance** characteristics of dependent systems and services

**DATA CONSTRAINTS** by **capturing real-world transactions from test and staging** environments and intelligently generating “life-like” virtual service and test automation assets to increase quality

**TESTING CONSTRAINTS** by **enabling multiple cross-functional team members**—including development and quality assurance—to rapidly design and execute automated unit, functional, regression, integration, load and performance tests

**VISIBILITY CONSTRAINTS** by **gaining transaction-level insight** into complex, modern applications, accelerating root-cause analysis of defects and remediation of discovered problems in intuitive ways

**As a result, IT teams can work in parallel and “shift left” development and test activities, so you can bring new and better services to market faster—and ultimately meet customer demand.**

# What's Next?

**In chapter 3** of the series, we'll look more closely at what it takes to achieve Agile Parallel Development and how it ties directly to your capabilities around virtualization, collaboration and software quality and management.

Are constraints holding you back? Discover your strengths and weaknesses in the **Develop & Test Assessment**, and receive actionable guidance you can use to start tackling your dev/test challenges today.

ca® technologies