Capacitas's Seven Pillars of Software Performance

Performance isn't just about speed...

You can't accurately measure performance using just a fraction of the information available. That's why we replaced traditional testing methods that only measure throughput and response time, with our *Seven Pillars of Software Performance*.

If any pillar fails, performance and end-user experience will suffer, potentially leading to lost revenue, increased costs, and unhappy customers.

Throughput & Response Time

Definition

Throughput: the rate at which requests are successfully completed

Response time: the time taken to successfully complete a request

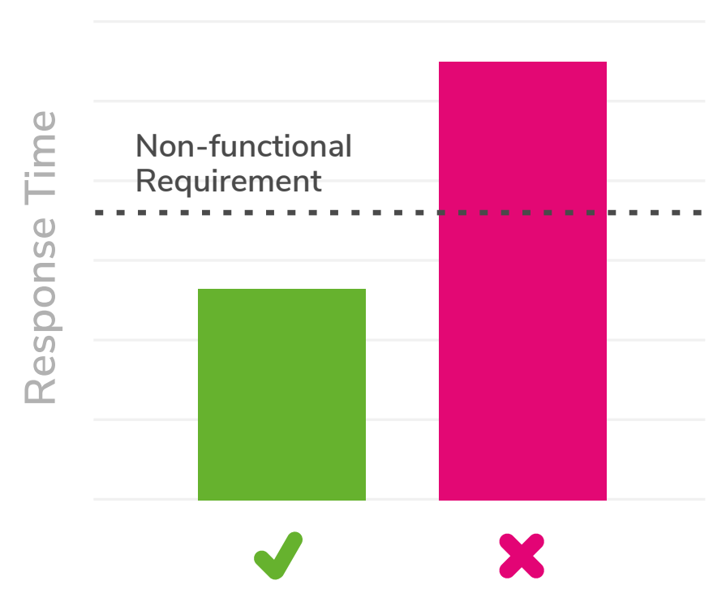
Business Relevance

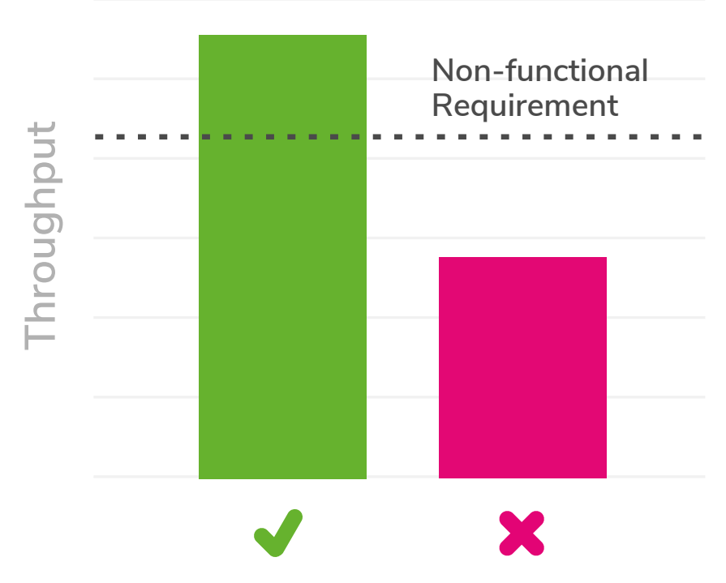
Response time drives conversion

How is it Measured?

- Web user experience monitoring tools
- APM tools
- Batch monitoring tools

What Does **Good** Look Like?





How Does it Relate to the Other Pillars?

- Consistently high response times can indicate software efficiency issues
- Inconsistent response times can indicate stability issues

Key Takeaways

- This pillar is the most widely analysed performance criterion
- Examining this pillar in isolation provides only narrow understanding of performance

Capacity

Definition

Capacity: the amount of resources required to support demand

Business Relevance

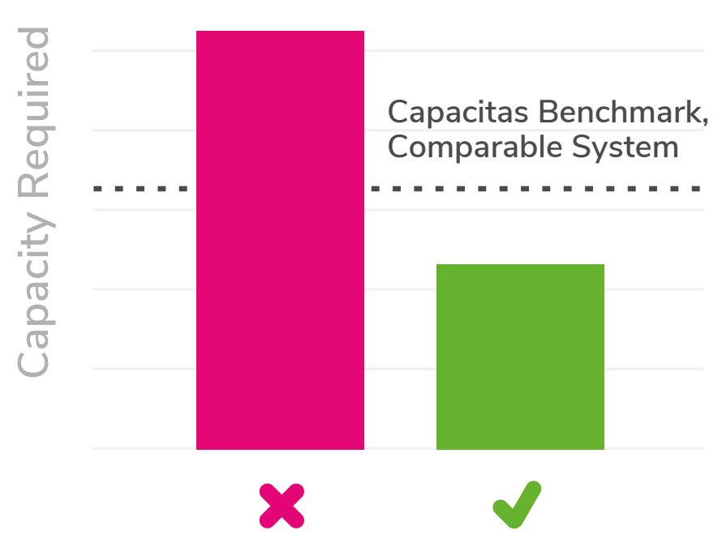
Capacity consumption drives cost of service

How is it Measured?

- The level of infrastructure (on premises or cloud) needed to support the service
- The utilisation of that infrastructure
- APM tools deliver great insight on throughput and response; they do not provide insight on capacity utilisation

What Does **Good** Look Like?

1. High utilisation with low queuing
2. The infrastructure footprint is similar to comparable systems



How Does it Relate to the Other Pillars?

- Insufficient capacity leads to high response times and constrained throughput
- Inefficient software creates excessive capacity consumption

Key Takeaway

- Although cloud infrastructure can automatically scale to meet demand, capacity management is still required to prevent costly inefficiencies

Efficiency

Definition

Efficiency: the amount of compute resource required to complete a business transaction, for example, the CPU time needed to search for a product in an e-commerce application

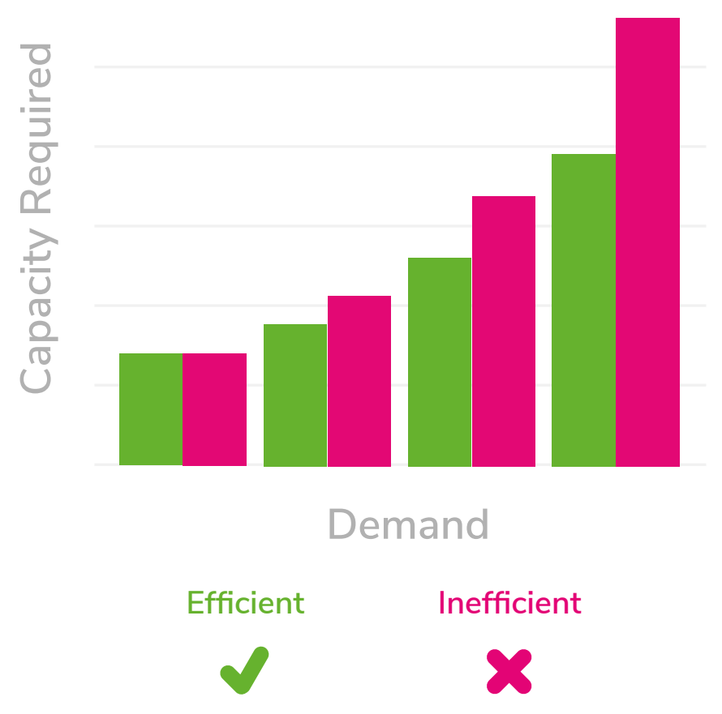
Business Relevance

Inefficient software increases cost of service

How is it Measured?

- By observing production or in test conditions

What Does **Good** Look Like?



How Does it Relate to the Other Pillars?

- Inefficiency can lead to increases in response times and excessive capacity consumption

Key Takeaway

- Software efficiency has a direct bearing on capacity consumption and thus cost

Scalability

Definition

Scalability: this is a measure of whether the software scales linearly with increasing demand and can use all the available capacity.

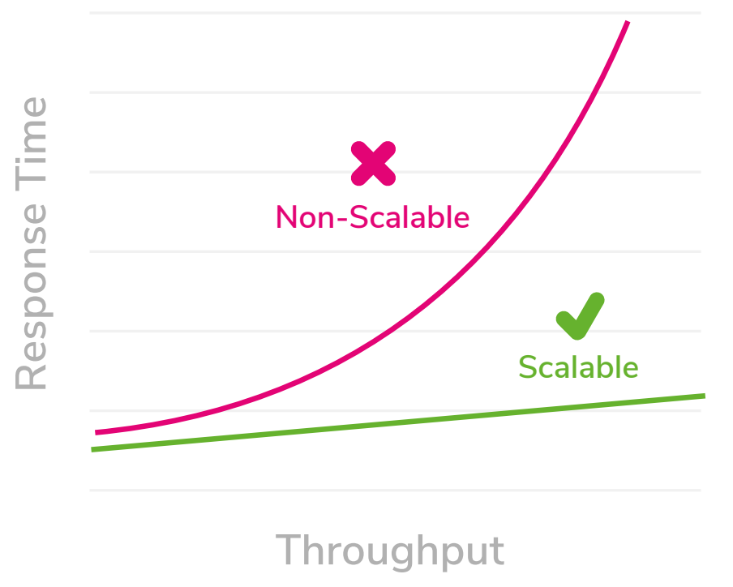
Business Relevance

Scalable software enables rapid business change

How is it Measured?

- The key metrics to measure are response time, throughput and utilisation. Measure utilisation across the service, e.g. server, network and application

What Does **Good** Look Like?



How Does it Relate to the Other Pillars?

- Non-scalable behaviour leads to degradation in response time and throughput. It can also impact software resilience

Key Takeaway

- If software isn't scalable it will act as a drag on the speed of delivering software change and decrease resiliency

Stability

Definition

Stability: This is a measure of the variation in response time and throughput over prolonged periods of load

Business Relevance

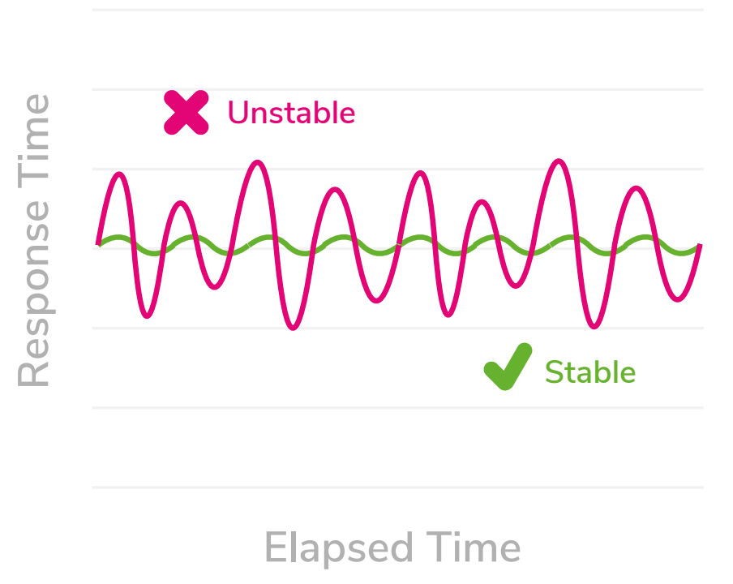
Stable software reduces cost of ownership and increases customer satisfaction

How is it Measured?

- The key metrics are response time, throughput and the utilisation of the platform that supports the software

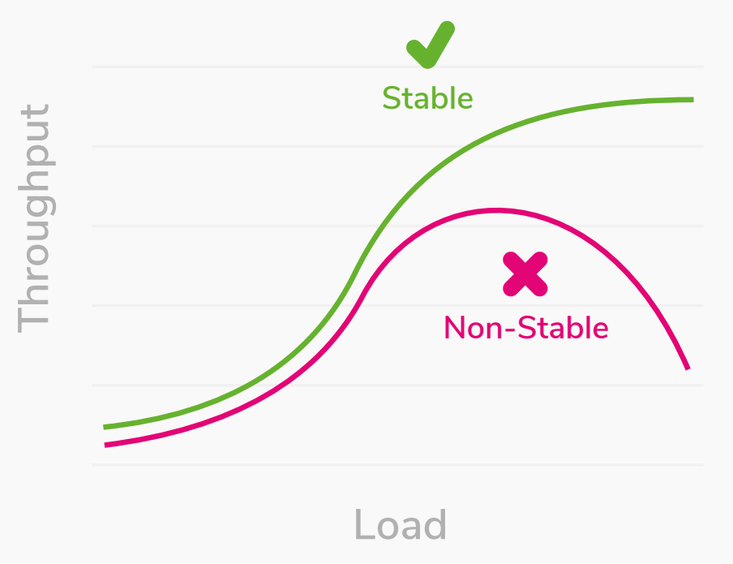
What Does **Good** Look Like?

1. Response times should be flat
2. Response time variability should be consistent
3. Key metrics follow a consistent pattern over prolonged periods



How Does it Relate to the Other Pillars?

- Stability relates to throughput, response time and resilience



Key Takeaway

- In an Agile/Continuous Delivery context, you need smart test analysis to detect instability pathologies

Resilience

Definition

Resilience: is how throughput and response time behave when an internal or external interface slows down or becomes unavailable

Business Relevance

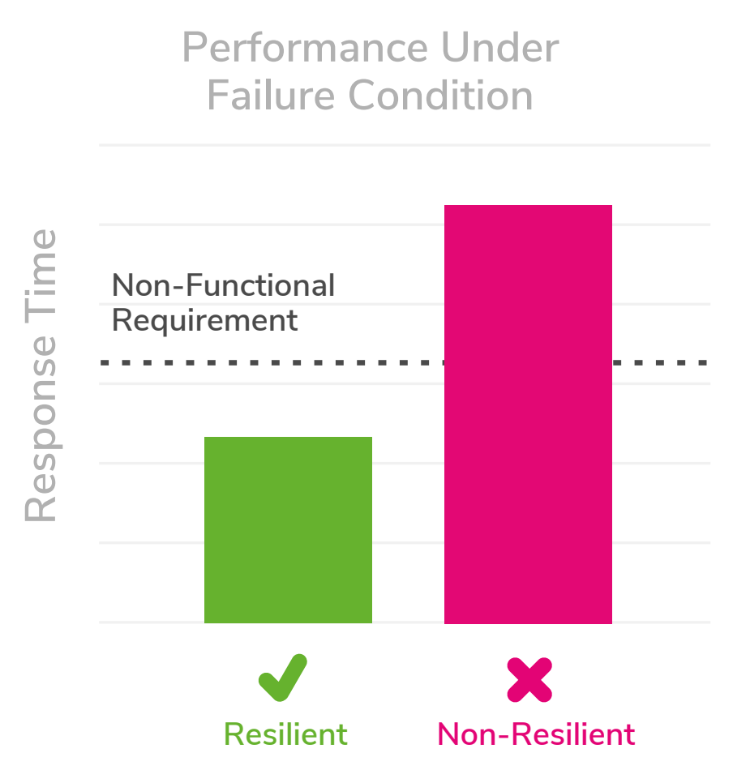
Resilient software reduces operational cost and risk of failure during peak periods

How is it Measured?

- The key metrics are response times and throughput of software components which do not directly call interfaces

What Does **Good** Look Like?

1. Overall response time and throughput should be unaffected by interface performance degradation



How Does it Relate to the Other Pillars?

- Poor resilience can lead to software instability

Key Takeaway

- The adoption of distributed software architectures (such as microservices) increases the risk of resilience issues

Instrumentation

Definition

Instrumentation: Ensuring you have a well-rounded tool set that empowers you to test across the other six pillars

Individually,**APM Tools** are an invaluable source of data, but without the right mix of tools you only achieve a very narrow understanding of software performance – therefore the Seventh Pillar is vital to your success.

Business Relevance

Good instrumentation promotes agile change and reduces the risk of service outage

What is Needed?

Including, but not limited to:

- Operating system monitoring
- APM tools
- Weblogs
- Network monitoring
- Application log files
- Database monitoring
- User experience monitoring

What Does **Good** Look Like?

1. The key metrics are collected and stored at the appropriate granularity
2. Coverage across the entire software stack
3. Coverage across business, service and component metrics
4. Consistent across all environments

How Does it Relate to the Other Pillars?

- Instrumentation provides the data required to assess the six other pillars

Key Takeaway

- Ensure instrumentation requirements are addressed at design stage to avoid expensive retrofitting

Key Takeaways

1. Performance is **not** simply about response times and throughput. That is too simplistic a way to measure performance. An all-embracing approach to measuring performance is required. *Capacitas's 7 Pillars of Software Performance (7PSP)* provide a comprehensive way of measuring performance.
2. Cloud services do **not** negate the need to manage capacity, costs and performance.
3. Consider **all** likely failure scenarios, not just business as usual.
4. Instrumentation requirements should be understood and addressed **early** in the software delivery lifecycle
5. If you're not designing in and testing for **all** 7 pillars you risk failing!

capacitas