Smart Development

This is **generic prerequisite** for the efficient development of software systems. It generally involves:

- Development discipline
- Continuous integration
- Unit-Driven Development
- Low-friction development
- Releasing early
- **Learning continuously**

For more ideas check out xLim articles or just Google up words like *Agile, XP, efficient development practices,* etc.

Messaging

Message-based systems are well-known for their capability to **decouple systems** and make them more **reliable against transient failures**. They require slightly different mental model as compared to blocking remote method calls, but it is worth it.

Enables:

- Implicit fault-tolerance
- System decoupling
- Asynchronous processing
- Basic scalability
- Better integration

CQRS

Command-Query Responsibility Segregation is a smart way of architecting software that uses messaging in a structured way. We separate **commands** (that change data) from **queries** (that read data) and differentiate between read and write sides.

Although this might not seem as a big advantage at this point, the real value is in the development options that are enabled by choosing this path.

Scalability Boost I

CQRS allows to achieve **better scalability** in systems without large development efforts or expensive hardware. We can efficiently **reduce the load** by distributing read data between multiple servers optimized for reads. We can also queue up command processing and **scale out** by splitting it between multiple workers.

Enables:

- Scaling out the read side
- Command queues
- Scaling command processing

Complexity Management

CQRS enables **Domain-Driven Design** by **capturing the intent** in form of commands.

DDD then helps to tackle the software development in *complex and rapidly changing business environment*.

Enables:

- Domain-Driven Design
- Repository Pattern
- Aggregate Roots
- Persistence Ignorance
- Ubiquitous Language

Reporting Power

CQRS **events are used to populate views** also known as **reporting side**. This approach creates decoupling between domain logic and reporting, allowing to change and optimize reports without altering other parts of the system.

Enables:

- Flexible Reporting
- Audit Logs
- Fast reports

Cloud Computing

Cloud Computing provides *computing, storage and network resources on-demand*. Resource consumers pay only for what they have used, no upfront investments are needed.

We can leverage cloud computing to respond to increased load by **scaling elastically** in response. Cloud infrastructure can also take care of scaling our read side or even distributing it around the globe.

Enables:

- Cloud Computing
- Elastic scaling
- Development Flexibility
- Reduced Costs

Cloud Computing Prices

Rackspace Cloud

Linux server: from $0.015 per hour (256 MB RAM, 10 GB HD)
Windows server: from $0.08 per hour (1024 MB RAM, 40 GB HD)
Store: from $0.15 / GB stored / month

Amazon EC2

Linux server: from $0.02 per hour (613 MB RAM, Cloud Store)
Windows server: from $0.03 per hour (613 MB RAM, Cloud Store)
Store: from $0.10 / GB stored / month

Windows Azure

Windows Role: starting from $0.12 per hour (1.7 GB RAM, 250GB HD)
Store: $0.15 / GB stored / month

Scalability II

By pushing further ideas of *Eric Evans*, *Greg Young* and *Pat Helland* further, we are capable to get our CQRS system closer to the concept of **almost-infinite scalability**. This **simplifies persistence and scaling**, while providing **more flexibility in tuning reliability and redundancy** options.

Enables:

- Almost-infinite scalability
- Reliability and redundancy tuning
- Scalability tuning
- Performance optimization
- Getting rid of 2PC and DTC

Business Value Boost

Event Sourcing allows to persist application entities as sequence of events that created them. This simplifies persistence and allows to **capture all business changes in lossless manner** (unlike usual CRUD or ORM).

Performance is not affected and we get native extension point for data mining and much better scalability.

Enables:

- Event Sourcing For Aggregate Roots
- Persisting all business data
- Specification Testing
- No longer need to use SQL

Business Data Mining

Event sourcing pushes CQRS reporting capabilities further. It allows to **define new reports** that provide **new insight** to the data that has already been captured. We can create **ad-hoc reports** and run **temporal queries** to look for trends, patterns and react to them.

Enables:

- New reports from the historical data
- ad-hoc reports
- Time-machine
- Behavioral analysis

Digital Nervous System

By pushing further ideas of people like *Eric Meyjer* (Cloud programmability for .NET) and *Bill Gates* (Digital Nervous System) while combining them with complex event processing we can bring our enterprise software to a much higher level of business value (and do all sorts of cool things).

Enables:

- Real-time notifications and alerts
- Flexible real-time dashboards
- Company-wide integration and automation with heuristics

Construction Bonus

We can distribute development effort of various CQRS parts between **specialized teams**. **Domain developers can focus** on areas, where their efforts matter the most, without worrying to much about reporting or performance.

Enables:

- Outsourcing UI to designers
- Native system APIs (commands + views)
- Outsourcing view denormalization
- Domain developers can focus on work