Orlando Code Camp

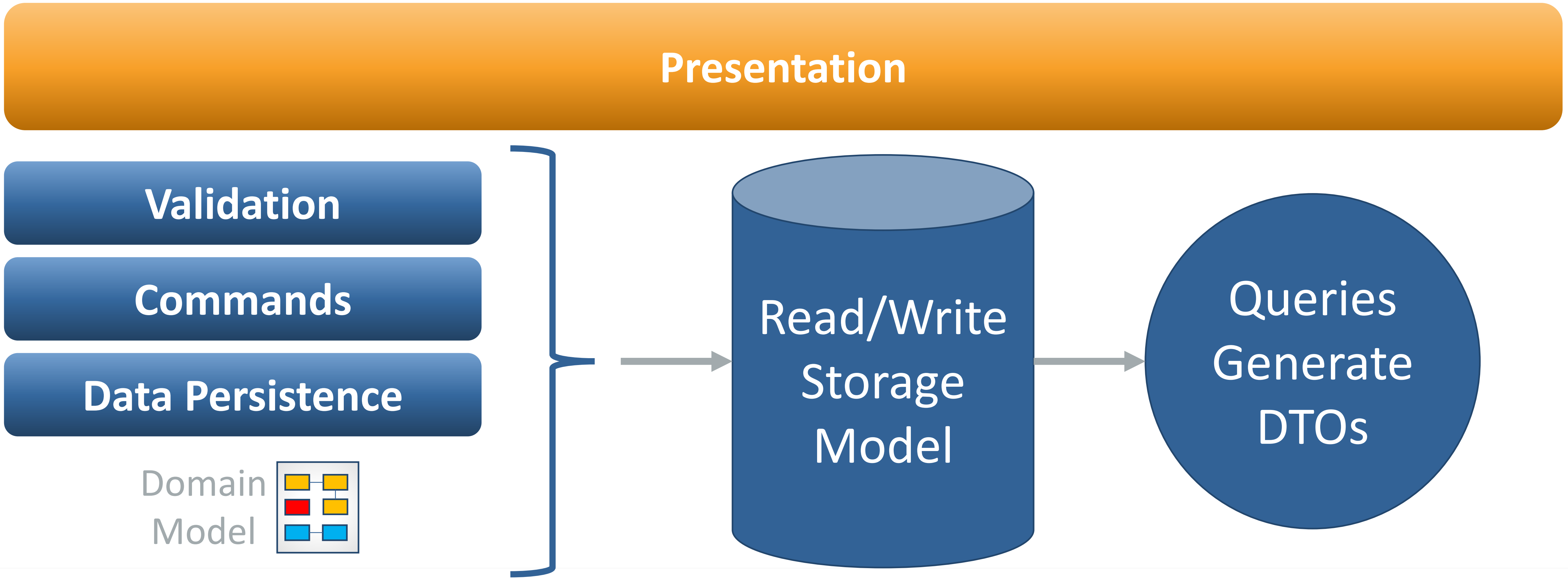
Command and Query Responsibility Segregation (CQRS) and Event Sourcing with .NET Core and Elasticsearch

Presented by
Jeremy Huckeba

agilethought
insightful solutions :: innovative technologies

First, a look at a traditional approach…

First, a look at a traditional approach



agilethought
insightful solutions :: innovative technologies

The challenges of a traditional approach

- Read and write operations and models are co-mingled
  - Records are treated as discreet entities to modify rather than a progression over time
- Complexity may increase by orders of magnitude with many additional data and read models
- Potentially results in complex relationships (relational SQL) or large nested documents (document databases)
- Read model queries can become complex
- Tracking changes to records over time requires additional effort*

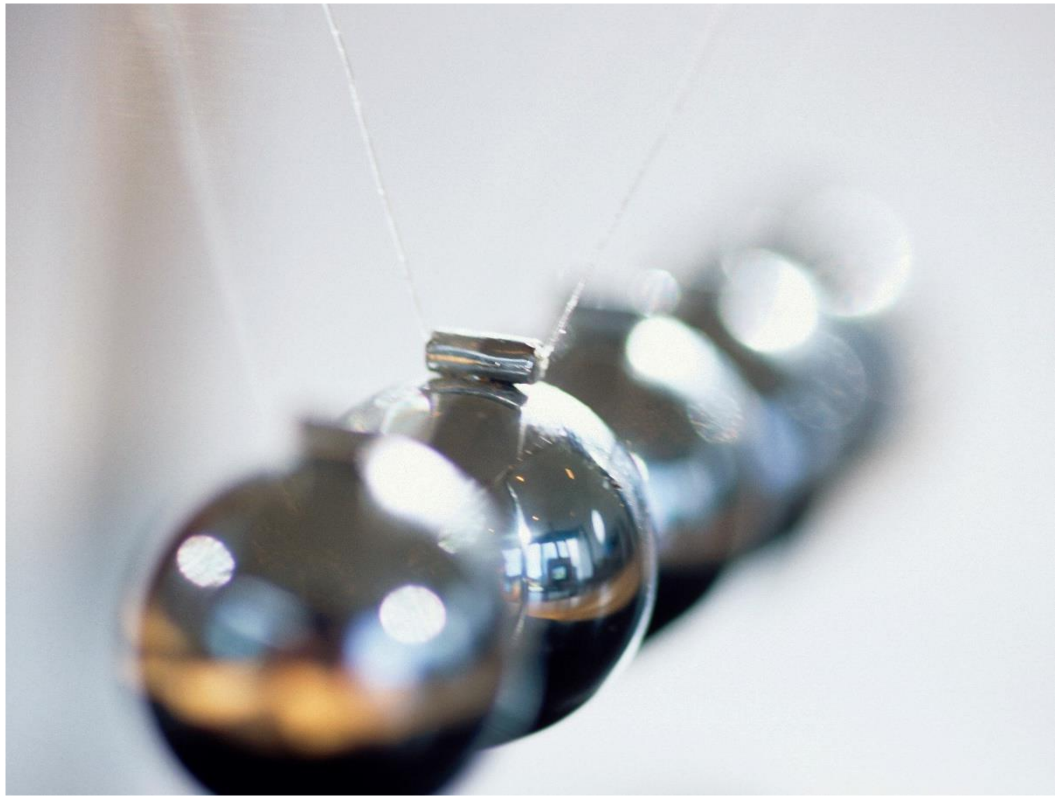
agilethought
insightful solutions :: innovative technologies

How is this pattern different?

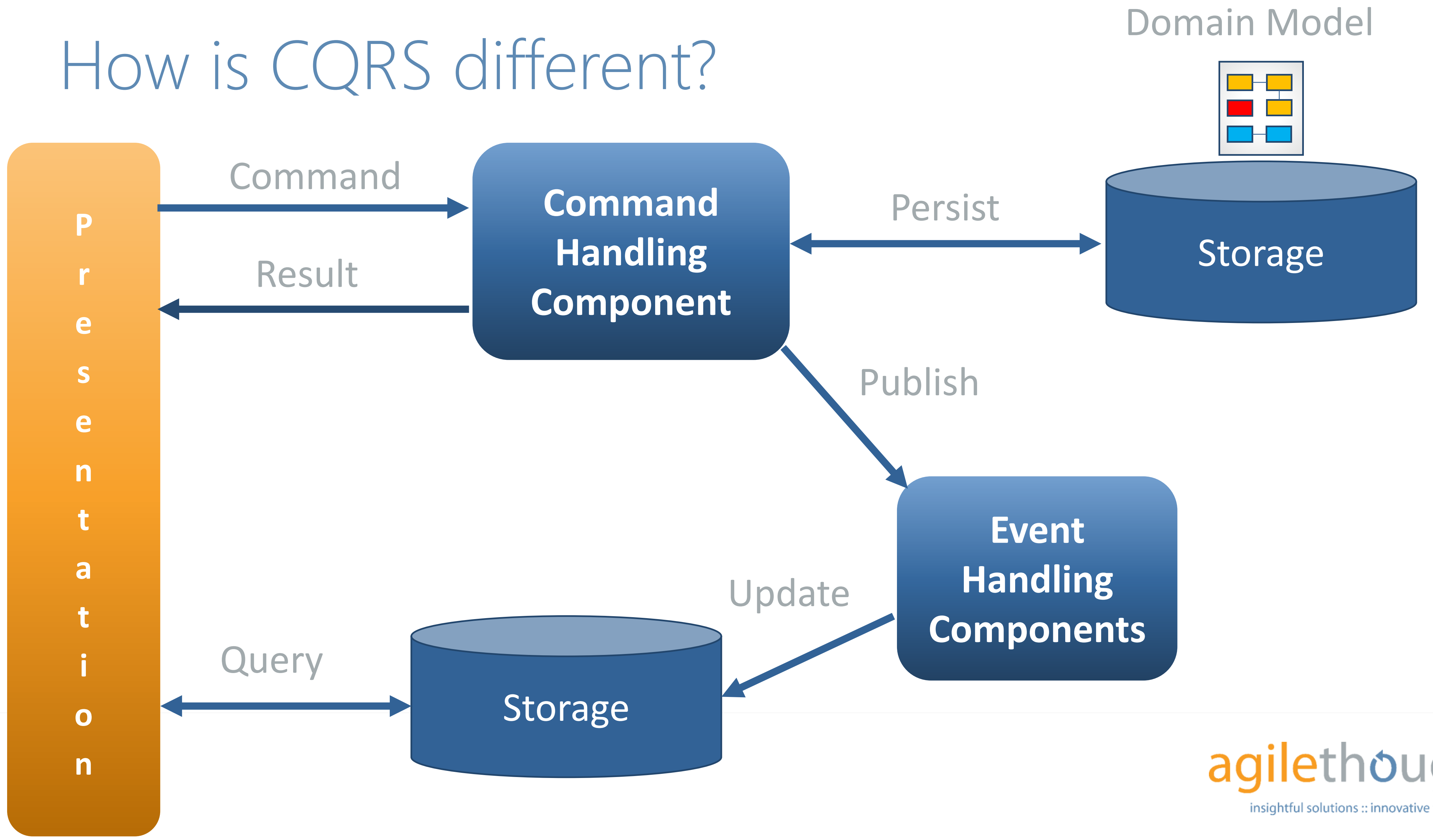
agilethought
insightful solutions :: innovative technologies

What is CQRS

- CQRS is a pattern that separates the responsibility of commands (do things) from that of queries (read)
- CQRS segregates operations that read data from operations that update data using separate models
- CQRS helps support the evolution of a system via separation of concerns
- CQRS works well with Domain Driven Design principles



How is CQRS different?



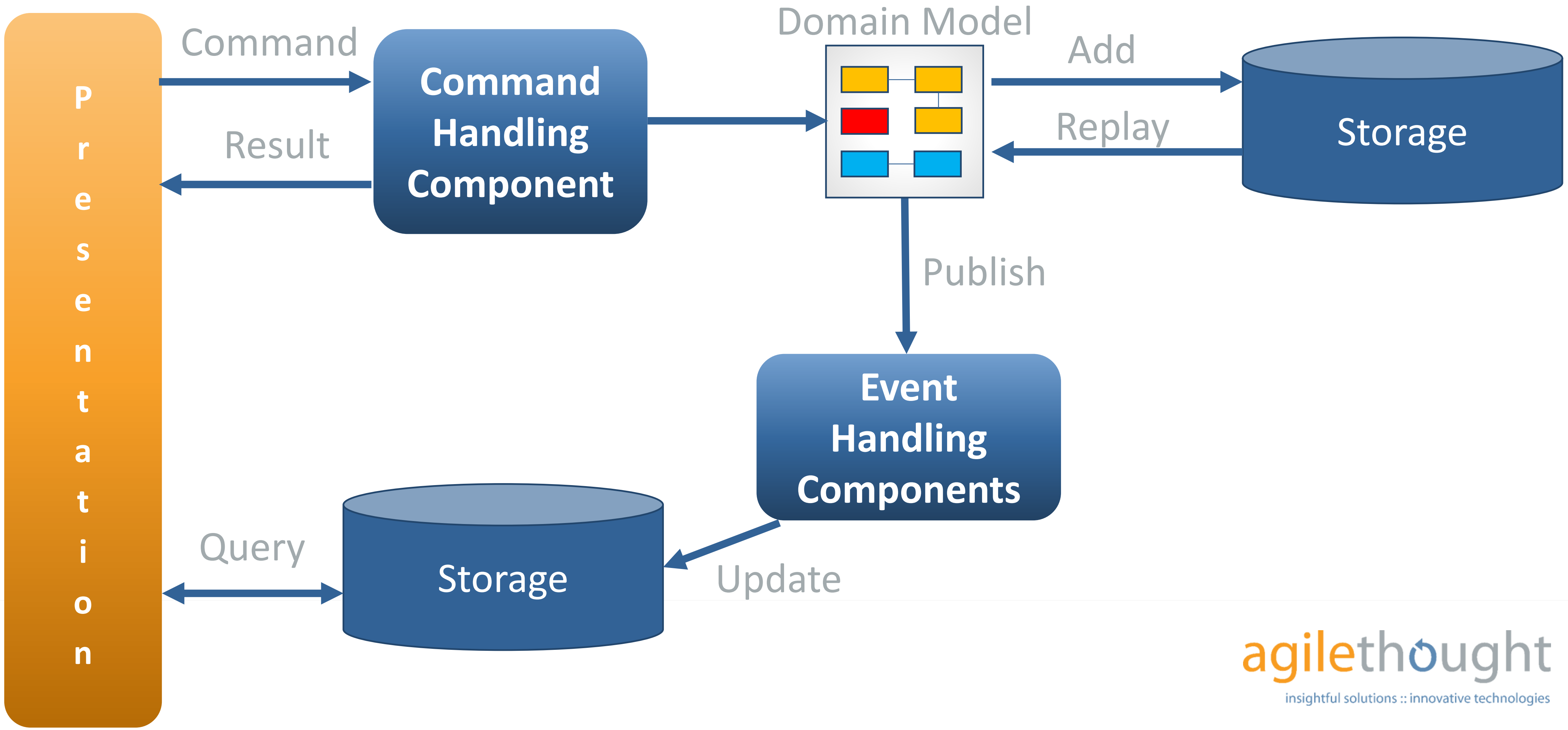
agilethought
insightful solutions :: innovative technologies

How does event sourcing fit in?

What is event sourcing?

- Event Sourcing is a pattern which ensures that all changes to application state are stored as a sequence of events
- Event Sourcing mirrors our perspective of time as a vector
- Event Sourcing allows us to know not just where we are, but how we got here
- We can restore a domain object by "replaying the tape" of events

Event sourcing works well with CQRS

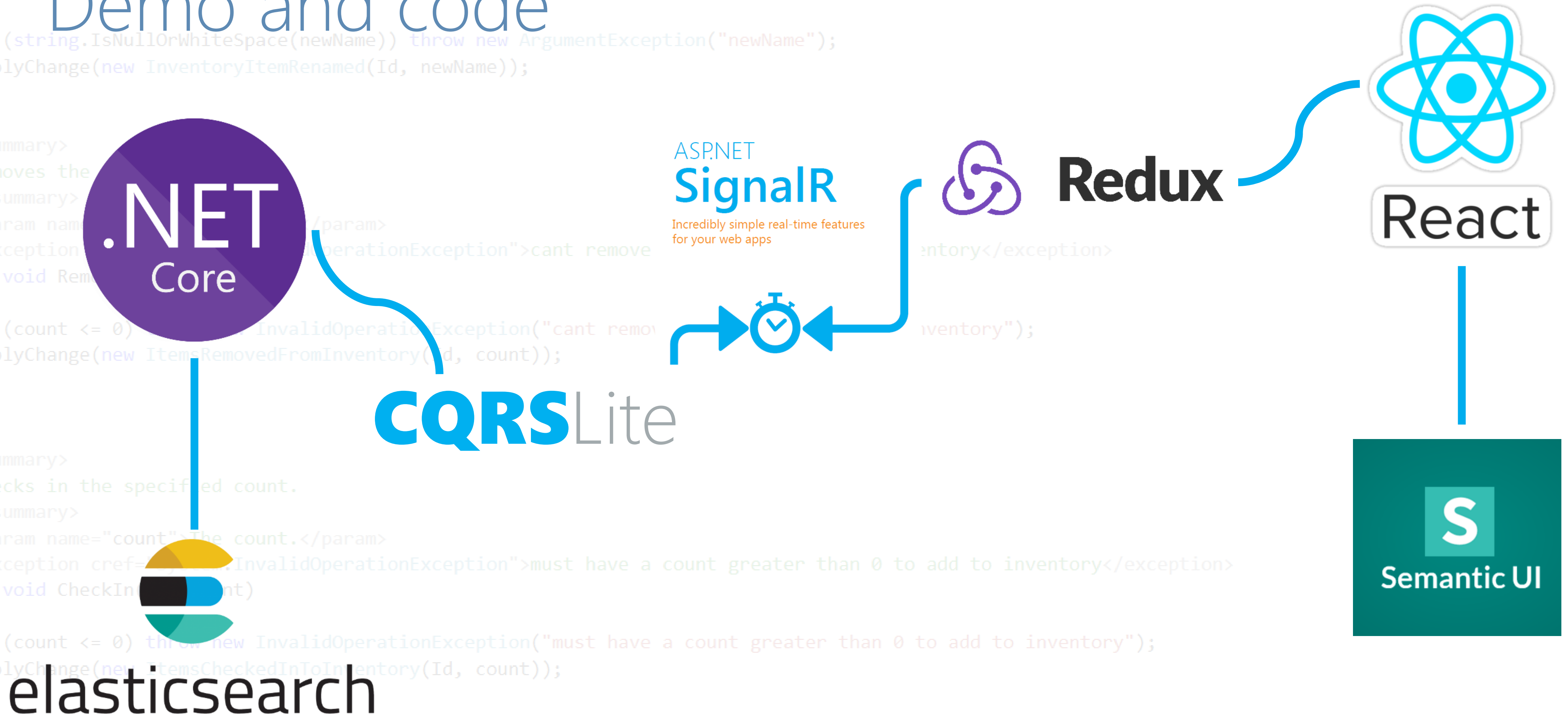


agilethought
insightful solutions :: innovative technologies

A “real-world” example

agilethought
insightful solutions :: innovative technologies

Demo and code



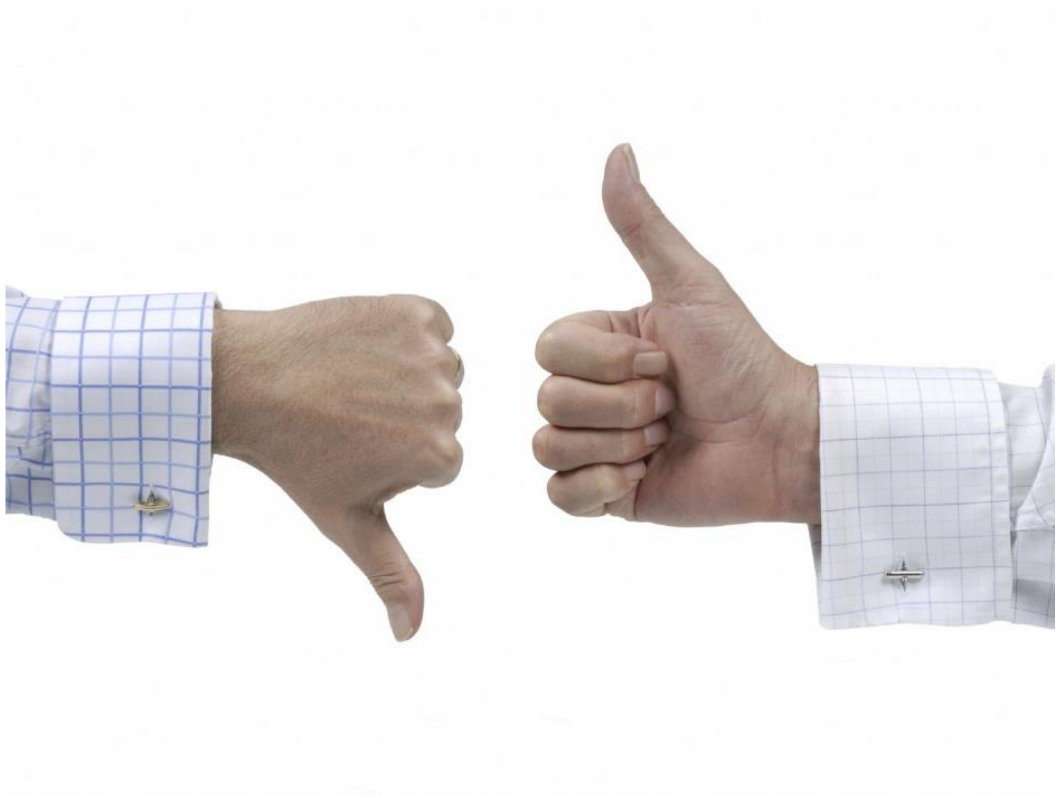
Pros and Cons

agilethought

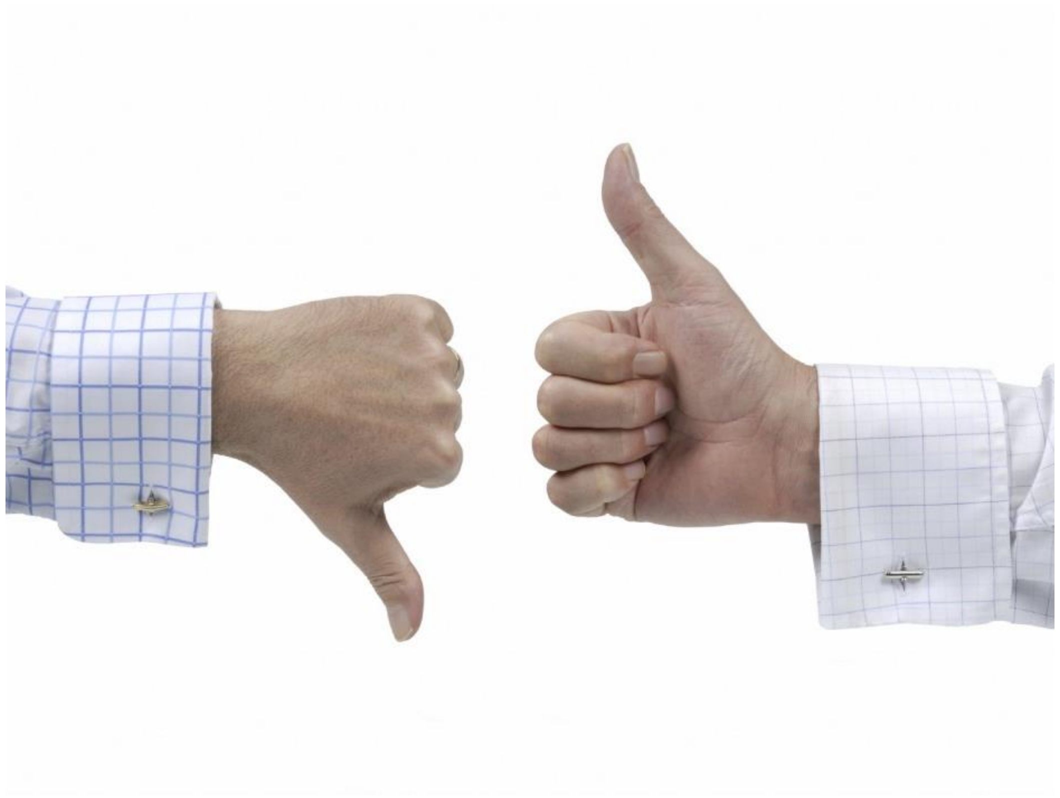
insightful solutions :: innovative technologies

All opposed?

- Can be overkill for simple line-of-business applications
- It takes more time to implement simple features and changes
- Time-to-market may increase due to additional engineering time, "but caveats"
- Some engineers are not familiar with the pattern
- Read and write models are only "eventually consistent“ with ElasticSearch
- Works best when there is a disciplined, domain-driven design effort prior to engineering efforts
- Can't be generated using scaffolding mechanisms (you have to code)



All in favor?



- Helps separate concerns in very complex applications
- Provides an auditable record of events over time
- Can be leveraged to provide advanced troubleshooting
- Makes it trivial to create views tailored to different needs and clients
- Allows for extremely performant and scalable read operations
- Forces developers account for concurrency
- Can secure different data views independent of write operations

agilethought

insightful solutions :: innovative technologies

Conclusion

agilethought

insightful solutions :: innovative technologies

Other resources

- github.com/gautema/cqrslite
- hub.docker.com/r/sebp/elk/
- www.elastic.co/guide/en/elasticsearch/reference/master/heap-size.html
- www.elastic.co/
- martinfowler.com/eaaDev/EventSourcing.html
- martinfowler.com/bliki/CQRS.html
- msdn.microsoft.com/en-us/library/jj554200.aspx
- Domain Driven Design (book)

Stay Connected

- linkedin.com/in/jshuckeba
- github.com/jshuckeba
- www.agilethought.com
- @AgileThought

If you have questions or would like more information, feel free to contact me via email jeremy.huckeba@agilethought.com

agilethought

insightful solutions :: innovative technologies