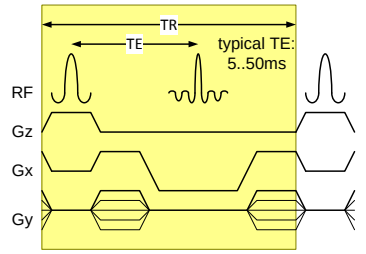
Multi-view Architecting; Illustrated by an MRI scanner

by *Gerrit Muller* University of South-Eastern Norway-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

Abstract

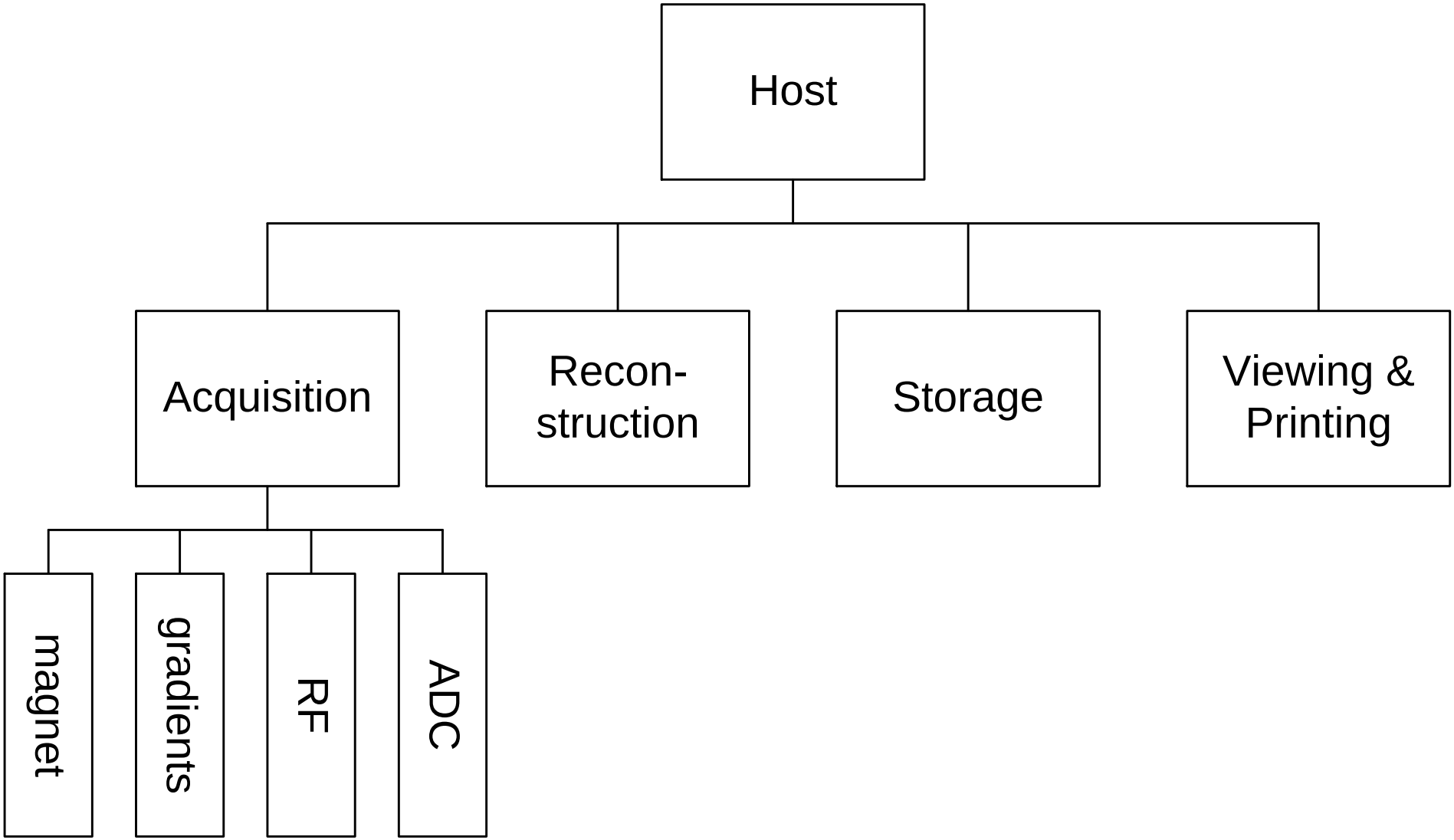
Many people expect from the system architect that he decomposes the system in smaller components and defines and guards the interfaces. The conventional waterfall model for software development and this architecture view form a dangerous combination: an extremely limited integral understanding with a very late feedback.
A multi-view architecting approach tackles the problem of integral understanding. In combination with spiral or incremental development models a powerful method becomes available for creating complex systems.



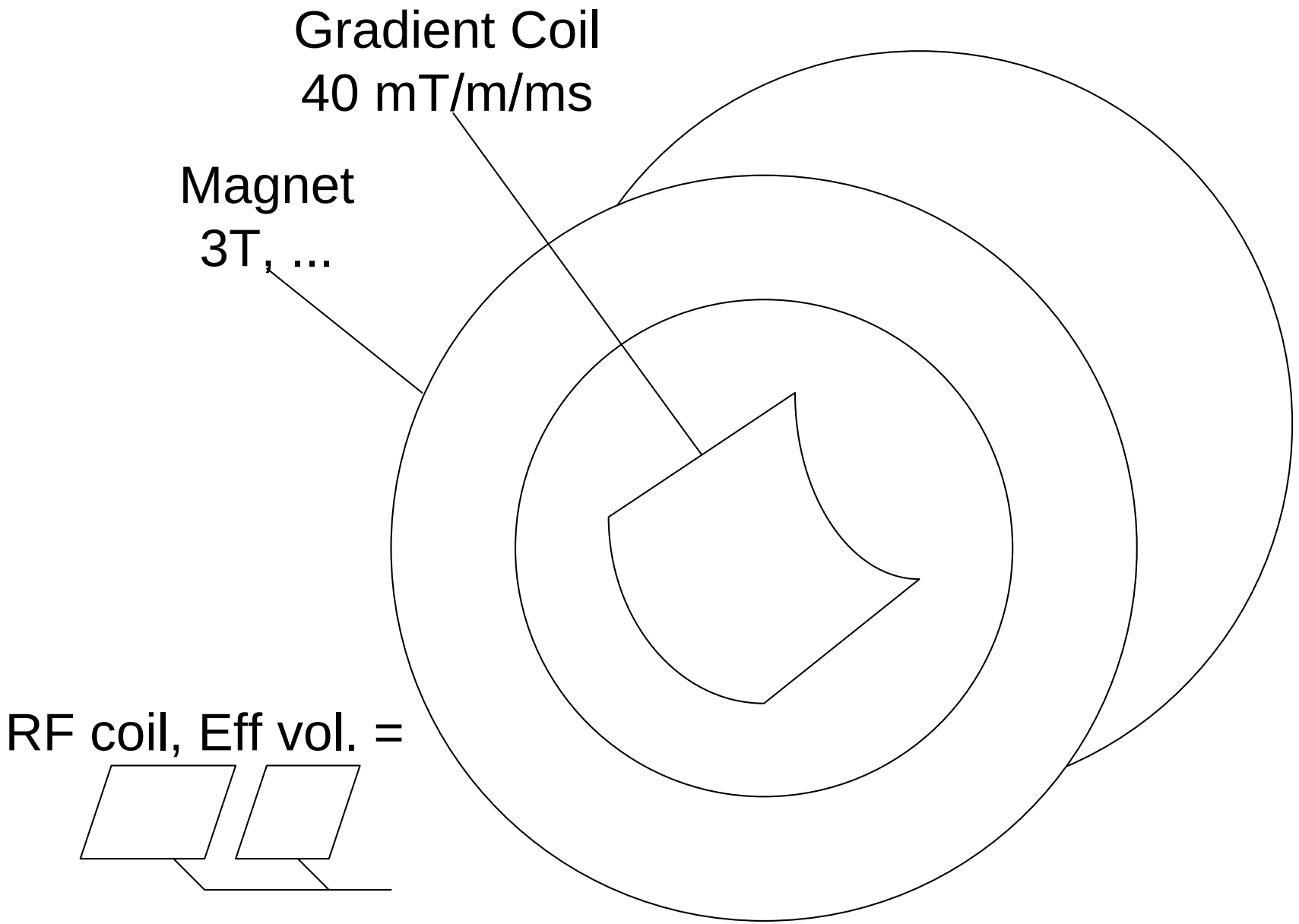
September 9, 2018
status: finished
version: 0.2

Illustration case: MRI scanner

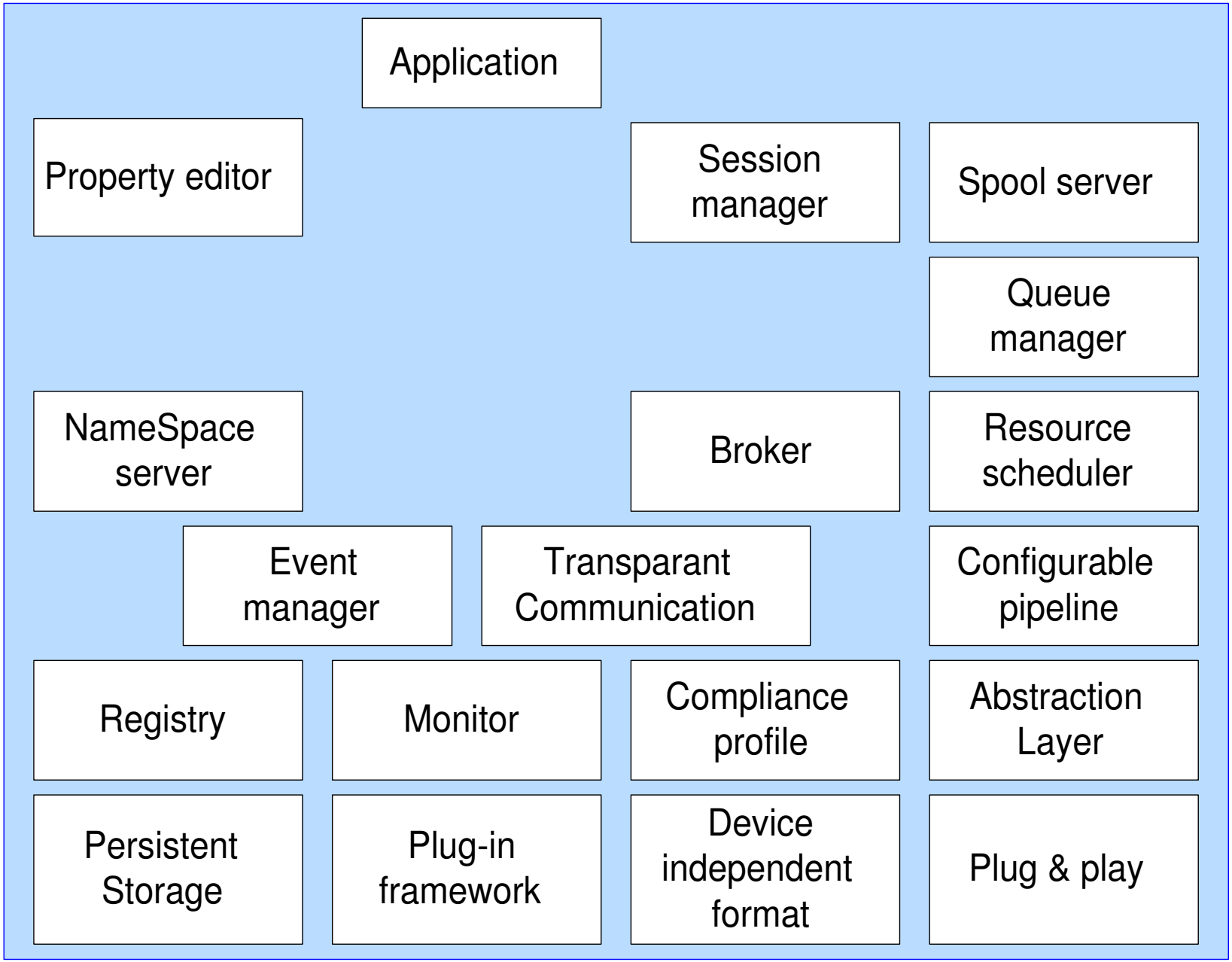
Block diagram view



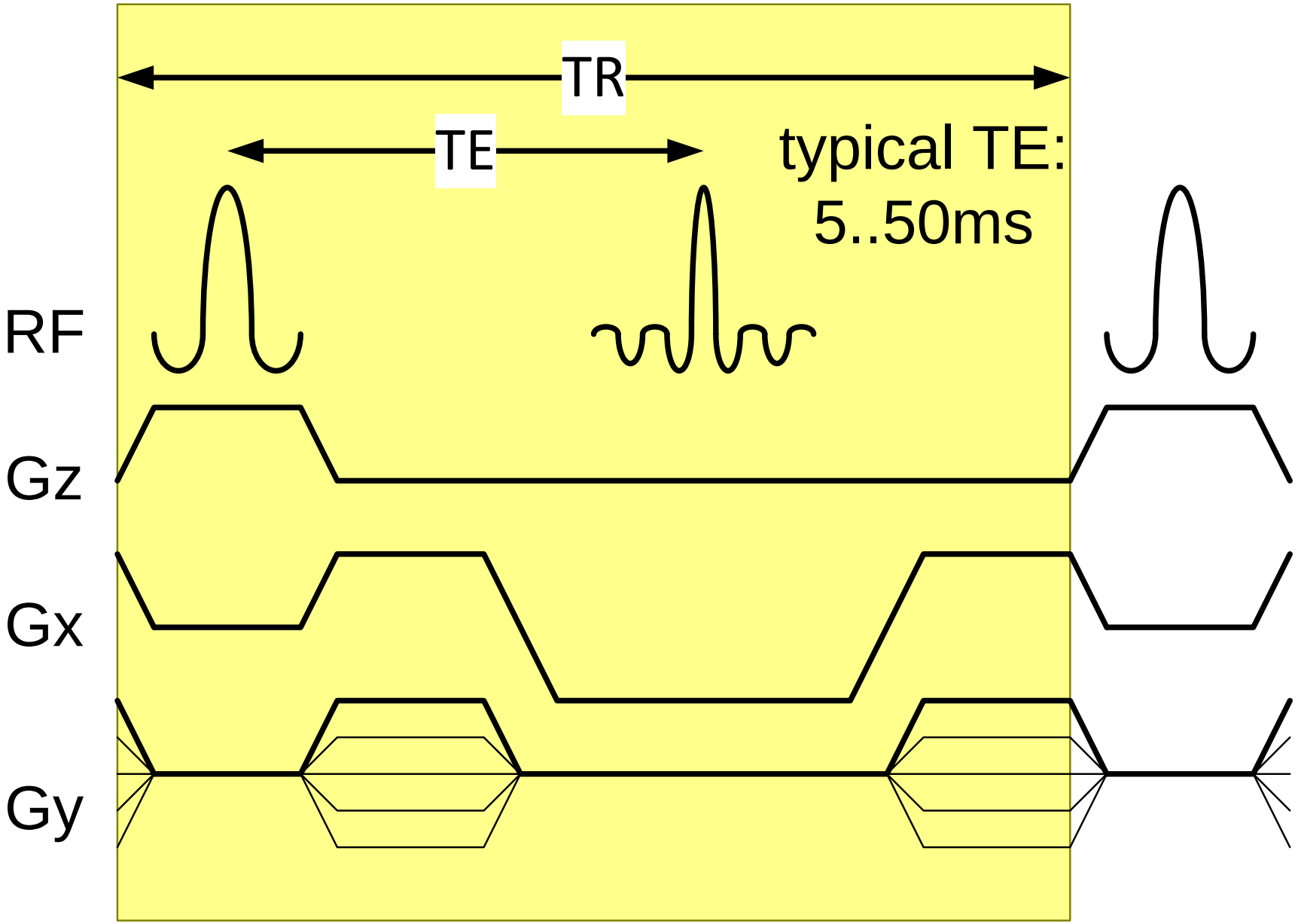
Physics view



Software architecture view



MR imaging methods view



Conceptual Work by the architect

- Most disciplines require multiple views, for instance circa 4 views in SW [Kruchten, Soni]
- Only a subset of disciplines has been shown (not shown are a.o. mechanics, logistics, project management)

The **system architect integrates** the **complementing disciplinary views**

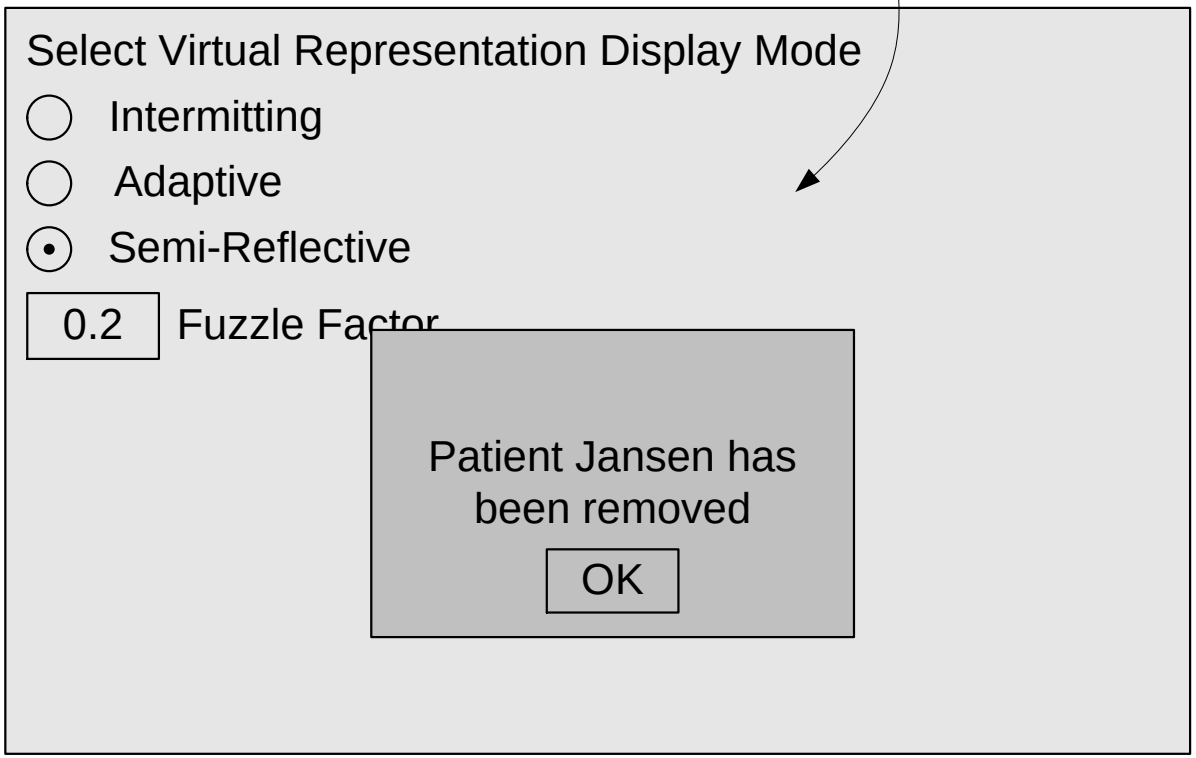
However

Decisions and trade-offs in the **conceptual view** are driven by **application**, **business** and **operational** inputs

Useability and main stakeholders

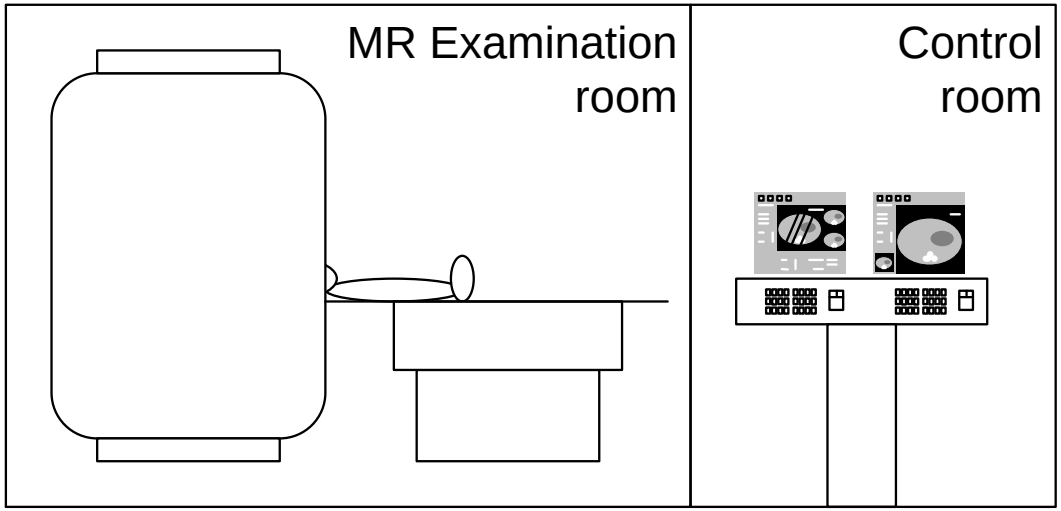
The engineer creates a technological UI...

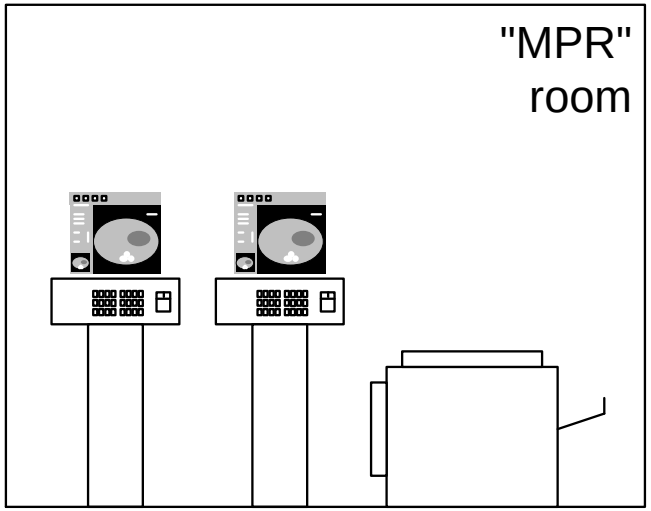
without imagining the clinical reality

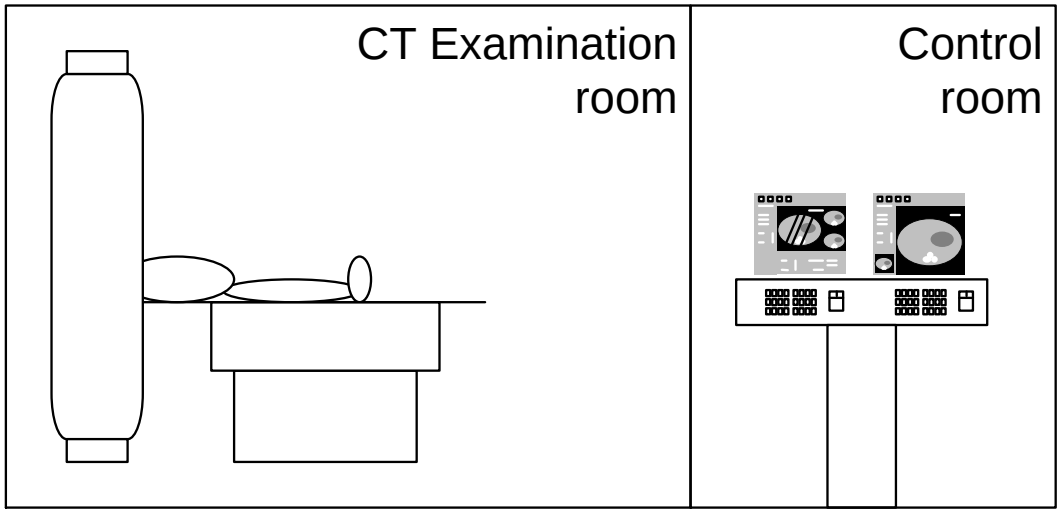


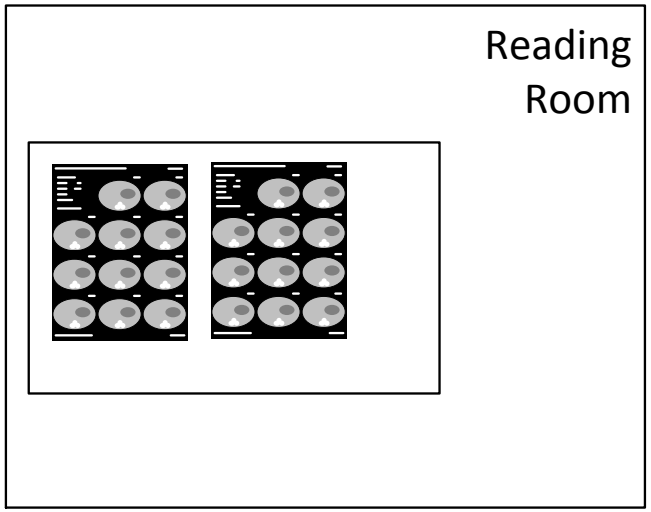
"In the meantime the patient is horrified by the intimidating system, the weird cage around his body and the EKG leads attached to his breast..."

Radiology department view

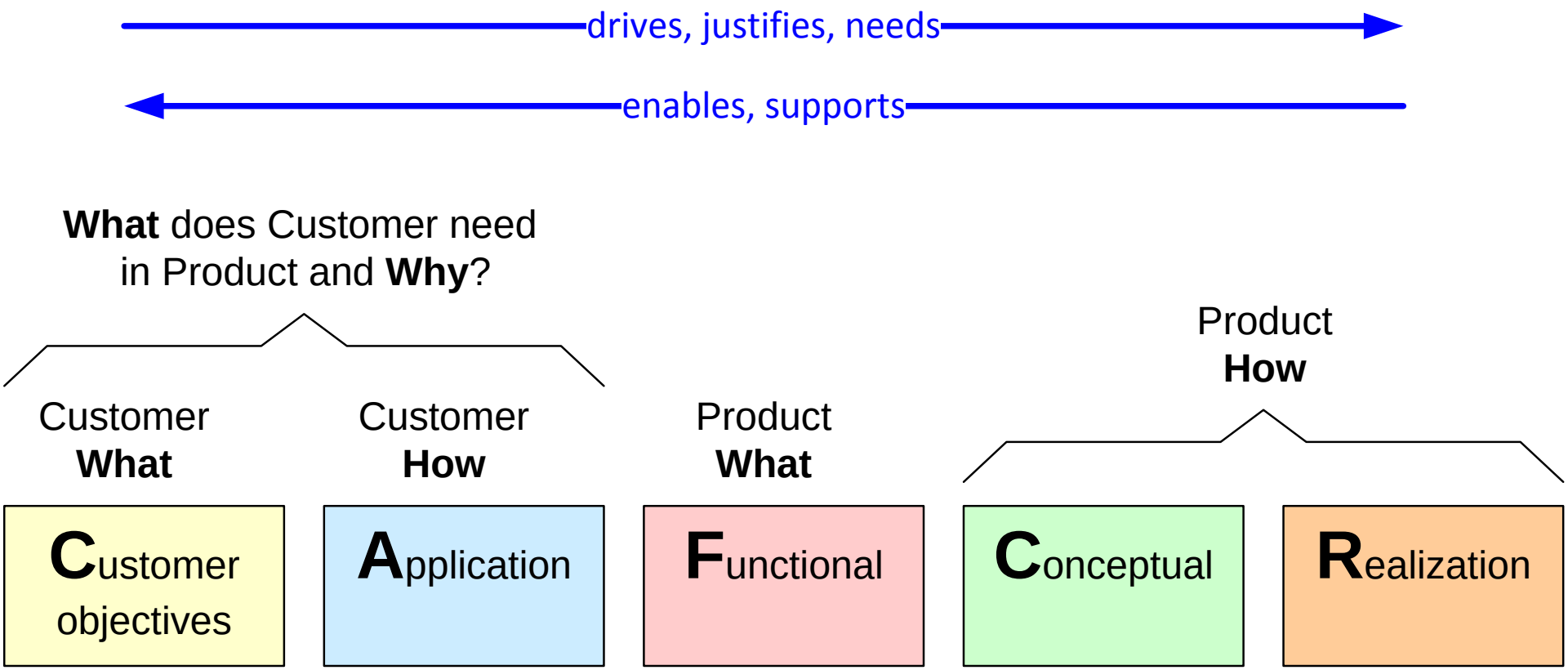




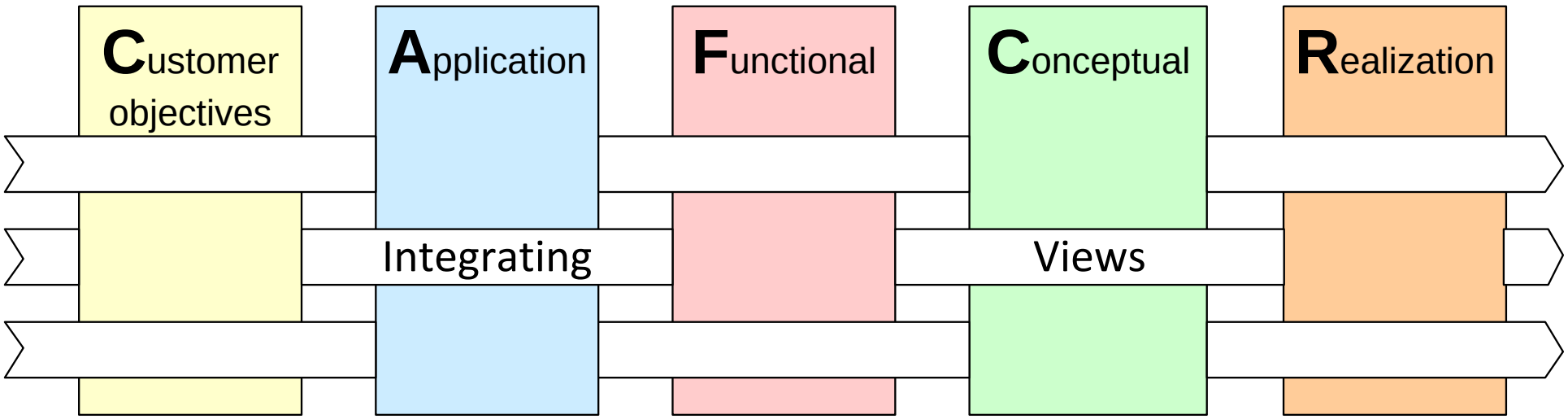




System Architect integrates 5 viewpoints



Integration of 5 views



High margin Cardiology market	Cost per examination model	Patient throughput features	System throughput model	Budget in seconds

From scenario to budget

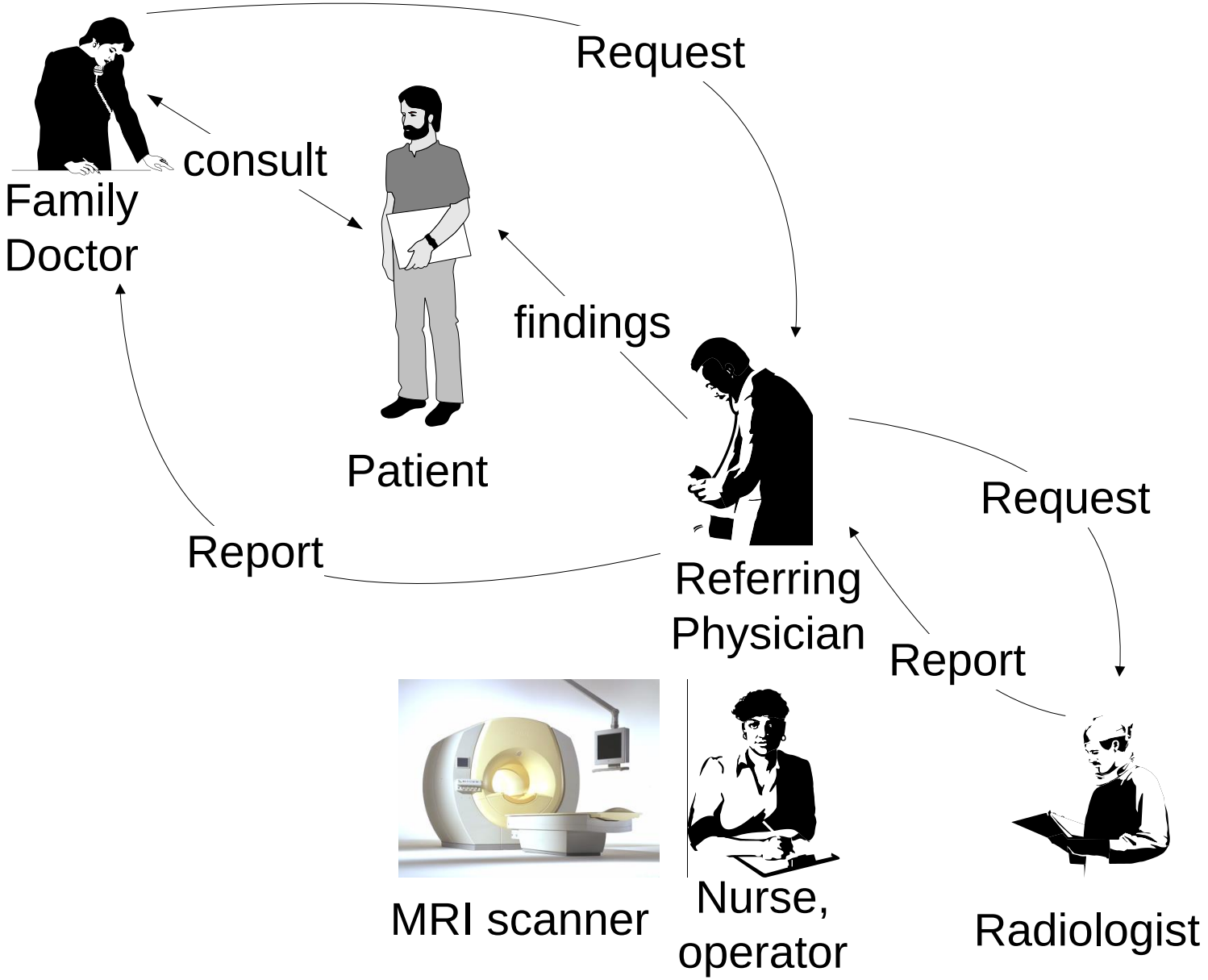
Customer objectives	**A**pplication	**F**unctional	**C**onceptual	**R**ealization
Scenario:		Typical Case:	Models	Technical estimates
Accessible story, clearly outlining a **frequently** occurring situation with a **valuable**, but **challenging** solution		Functions and Quantification of **frequently** occurring **important** and **critical** case	Functional and Performance	

Several iterations are required. In later iterations worst cases and exceptional cases are taken into account. The technical estimates are then transformed in budgets.

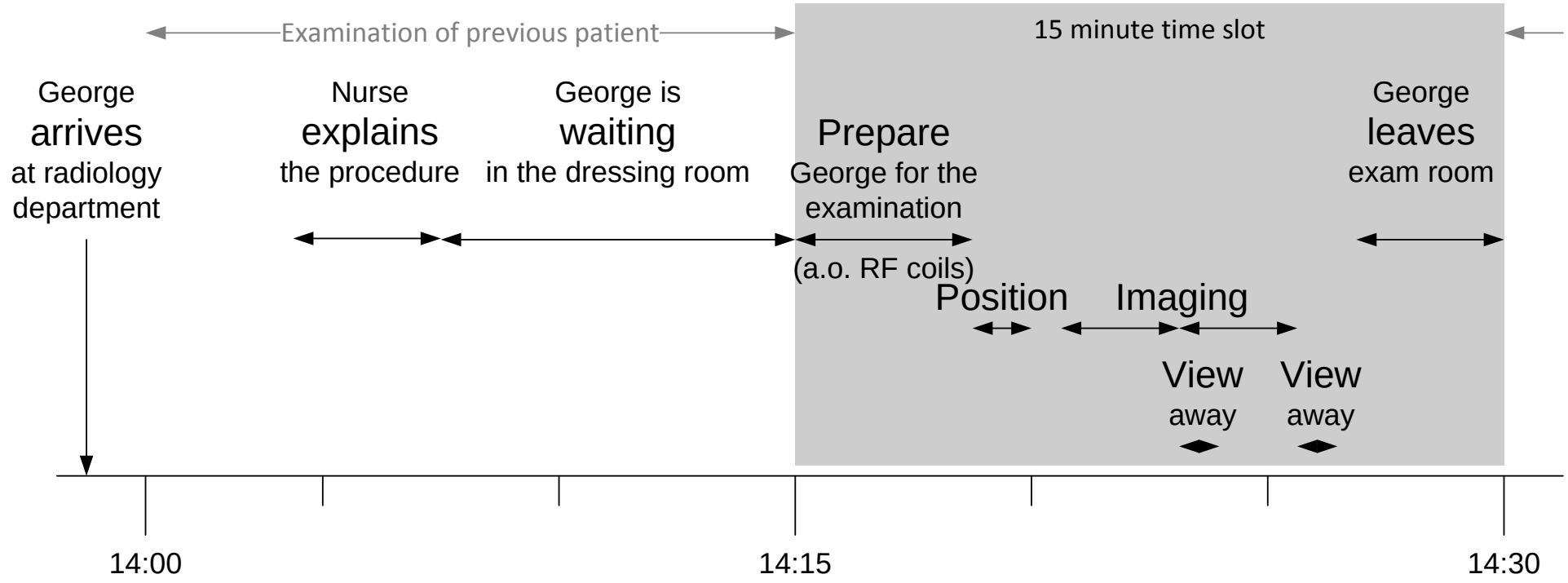
MR neuro scenario

- Patient George has continuous headache.
- His family doctor has send him to the Neurologist.
- The Neurologist wants to exclude the possibility of a tumor and requests an MRI examination.
- The Radiologists does not see any indication for a tumour.
- The Radiologist sends his report to the Neurologist.
- The Neurologist discusses his findings with the patient and sends a report to the family doctor.

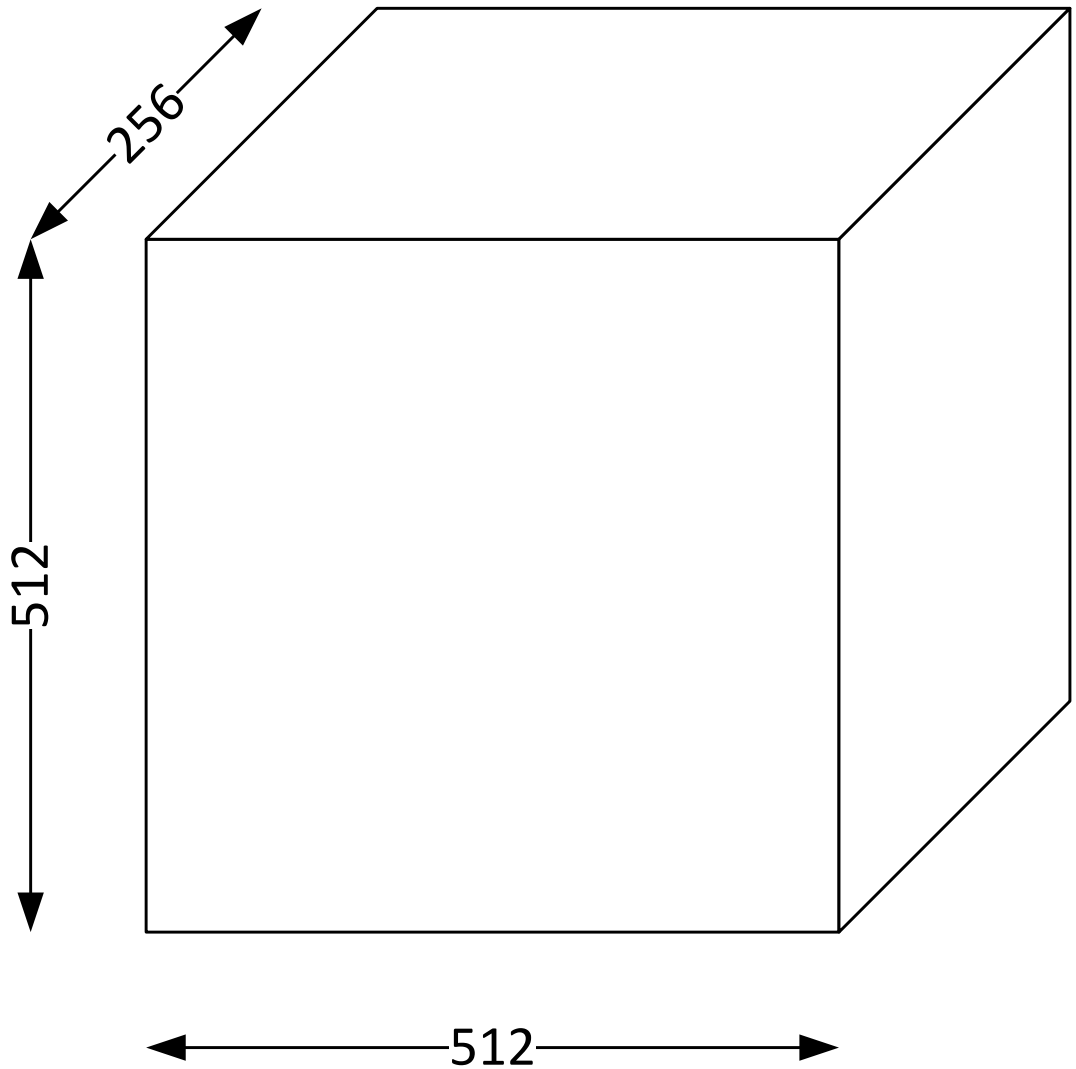
Clinical Stakeholders



Typical timing of Neuro examination



Typical amount of Images: 2 Volumes



Data in bytes =

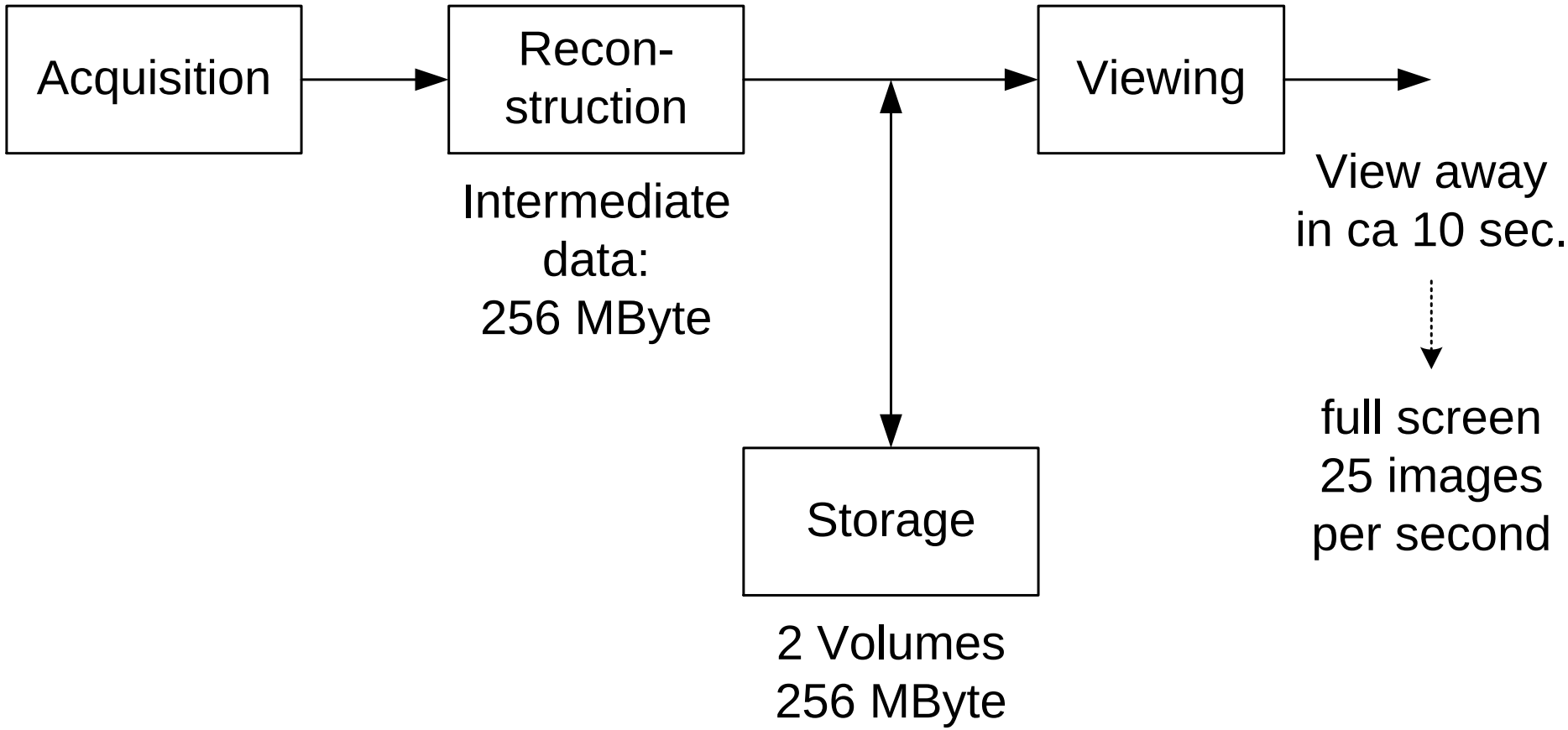
2 * 512 * 512 * 256 * 2 =

Volumes x y z bytes per pixel

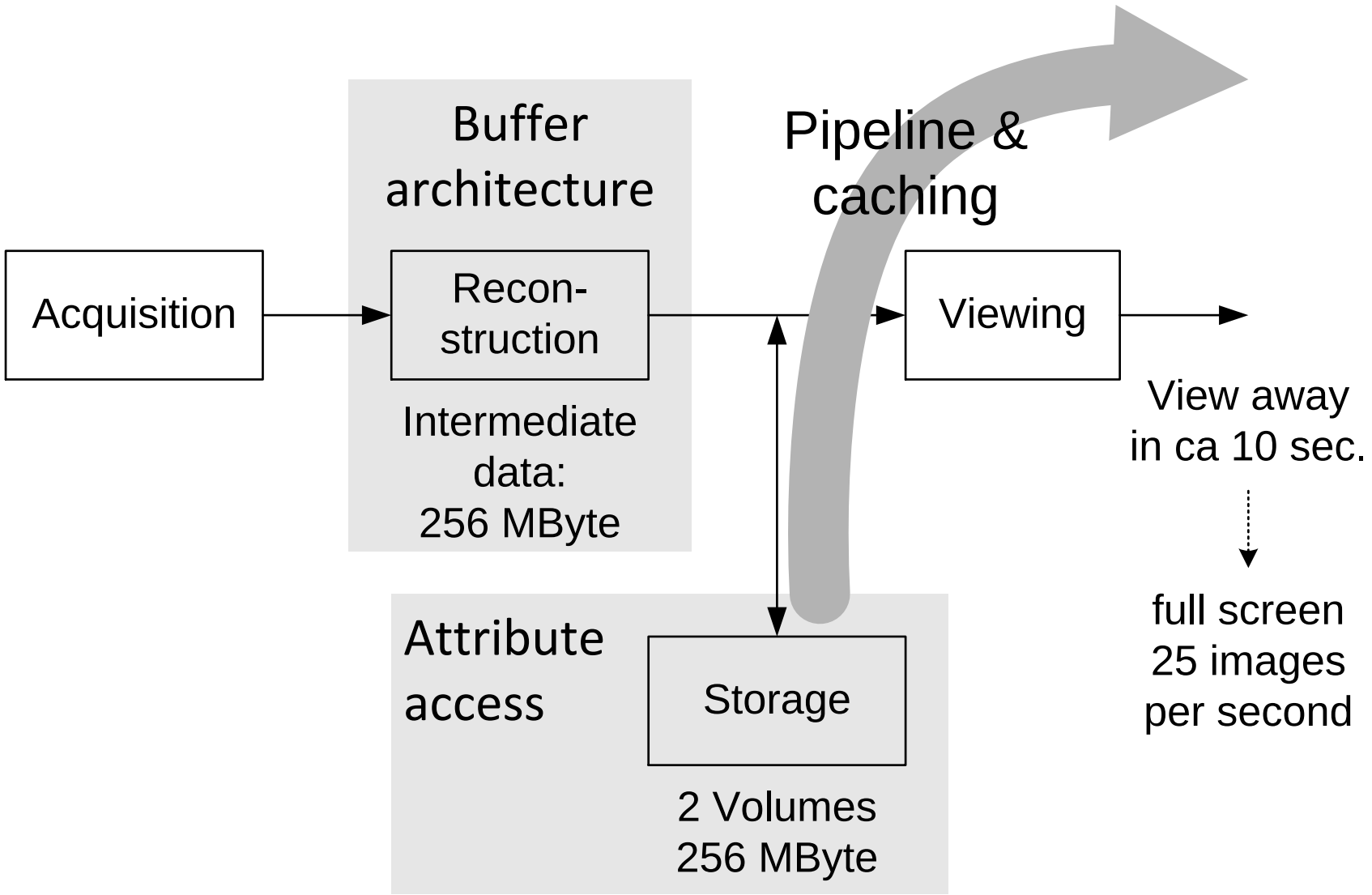
256 MBytes

in 2 * 2 minutes =
240 seconds

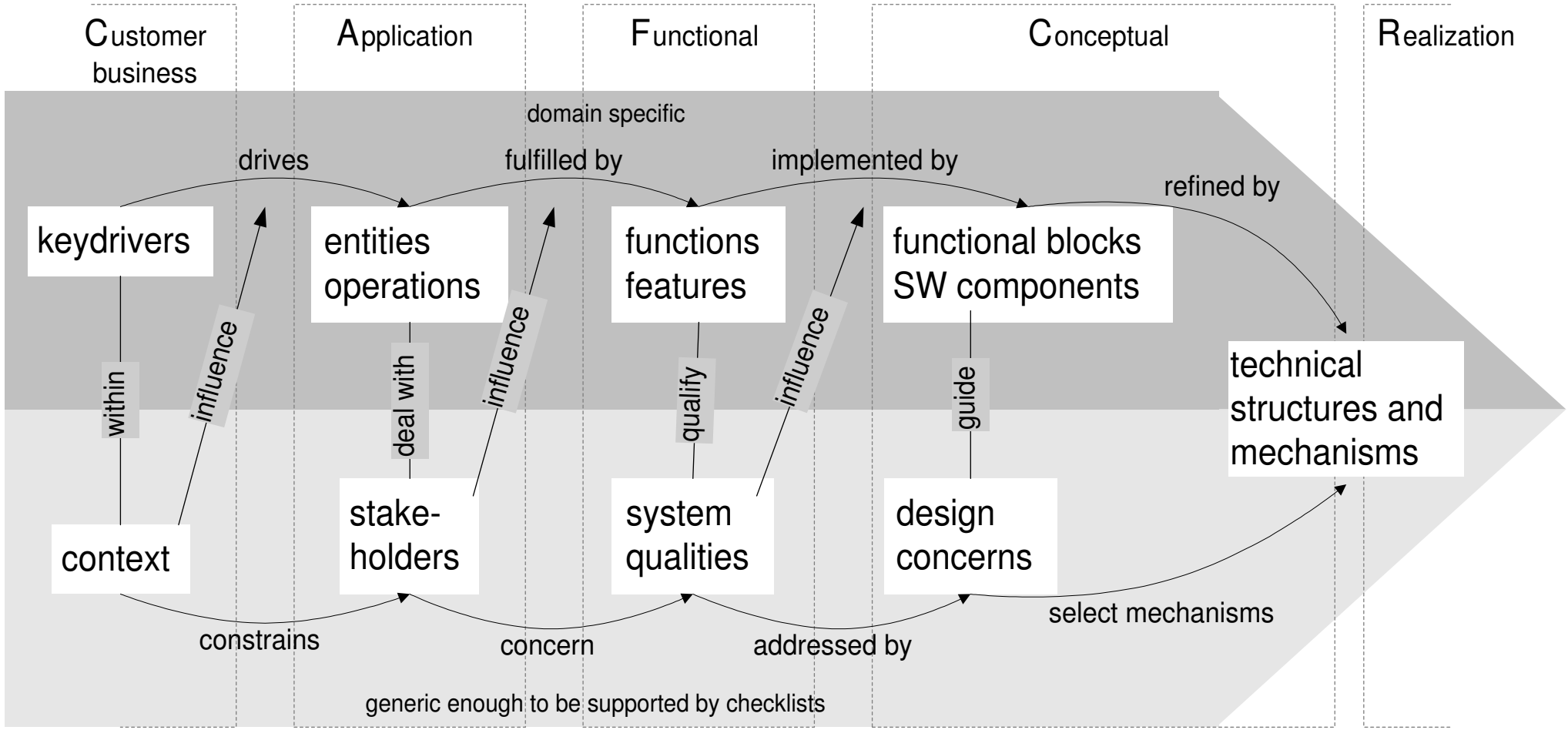
MR resource model



MR critical design choices



Checklists for integrating 5 views



Actual checklists

Customer business

Who decides?
Who pays?

Application

Consumer
User
Operator
Retailer

Safety	Response Time
Security	Image Quality
Reliability	Reproduceability
Robustness	Predicatability
Manufacturability	Accuracy
Testability	Cost price
Serviceability	Cost of operation
Configurability	Interaction with environment
Installability	Power consumption
Evolvability	Consumption rate (water, air, chemicals, etcetera)
Portability	
Upgradeability	Disposability
Extendability	Size, weight
Maintainability	Resource utilization
Useability	Logistics flexibility
Appeal, Appearance	Lead time
Throughput or Productivity	Standards Compliance

Functional

Operational Stakeholders:
Sales person
Field service engineer
Marketeer
Portfolio manager
Project manager
Developer

Conceptual

Granularity, Scoping, Containment, Cohesion, Coupling
Interfaces, Allocation, Bugets
Information model (entities, relations, operations)
Identification, Naming
Static characteristics, Dynamic behavior
System level infrastructure
Software development process, Environment, Repository, Tools
Feedback tools (for instance monitoring, statistics and analysis)
Persistence
Licensing, SW-keys
Set-up sequence
Technology choices
Make, Outsource, Buy, or Interoperate decisions
Meta-functional aspects:
- Operational (e.g. image processing, handling calls,…)
- Initialization, Start-up, Shutdown
- Fault handling
- Diagnostics
- Configuration handling
- Data replication
- Performance observation
- Capability query
- Testing
- Debugging
- Off-line guidance

Realisation

Exceptions, Logs, Traces
Process, Tasks, Threads
Configuration management: Packages, Components, Files, Objects, Modules, Interfaces
Automated testing: Special methods, Harness, Suites
Signalling, Messaging, Call-back scheduling, Notification, Active data, Watchdogs, Time-outs
Locking, Semaphores, Transactions, Checkpoints, Deadlock detection, Roll-back
Identification, Naming, Data model, Registry, Scoping, Configuration database, Inheritance
Resource management, Allocation, Fragmentation prevention, Garbage collection
Persistence, Caching, Versioning, Prefetching, Lazy evaluation
Licensing, SW-keys,
Bootstrap, Discovery, Negotiation
Call graphs, Message tracing, Object tracing
Distribution, Allocation, Transparency; Component, Client/Server, Multi-tier model

Coverage of MR neuro view

Customer business

Who decides?
Who pays?

Application

Consumer
User
Operator
Retailer

Functional

Safety
Security
Reliability
Robustness
Manufacturability
Testability
Serviceability
Configurability
Installability
Evolvability
Portability
Upgradeability
Extendability
Maintainability
Useability
Appeal, Appearance
Throughput or Productivity

Response Time
Image Quality
Reproduceability
Predicatability
Accuracy
Cost price
Cost of operation
Interaction with environment
Power consumption
Consumption rate (water, air, chemicals, etcetera)
Disposability
Size, weight
Resource utilization
Logistics flexibility
Lead time
Standards Compliance

Operational Stakeholders:
Sales person
Field service engineer
Marketeer
Portfolio manager
Project manager
Developer

Conceptual

Granularity, Scoping, Containment, Cohesion, Coupling
Interfaces, Allocation, Budgets
Information model (entities, relations, operations)
Identification, Naming
Static characteristics, Dynamic behavior
System level infrastructure
Software development process, Environment, Repository, Tools
Feedback tools (for instance monitoring, statistics and analysis)
Persistence
Licensing, SW-keys
Set-up sequence
Technology choices
Make, Outsource, Buy, or Interoperate decisions
Meta-functional aspects:
- Operational (e.g. image processing, handling calls,…)
- Initialization, Start-up, Shutdown
- Fault handling
- Diagnostics
- Configuration handling
- Data replication
- Performance observation
- Capability query
- Testing
- Debugging
- Off-line guidance

Realization

Exceptions, Logs, Traces
Process, Tasks, Threads
Configuration management: Packages, Components, Files, Objects, Modules, Interfaces
Automated testing: Special methods, Harness, Suites
Signalling, Messaging, Call-back scheduling, Notification, Active data, Watchdogs, Time-outs
Locking, Semaphores, Transactions, Checkpoints, Deadlock detection, Roll-back
Identification, Naming, Data model, Registry, Scoping, Configuration database, Inheritance
Resource management, Allocation, Fragmentation prevention, Garbage collection
Persistence, Caching, Versioning, Prefetching, Lazy evaluation
Licensing, SW-keys,
Bootstrap, Discovery, Negotiation
Call graphs, Message tracing, Object tracing
Distribution, Allocation, Transparency; Component, Client/Server, Multi-tier model

Architects must increase customer side contribution