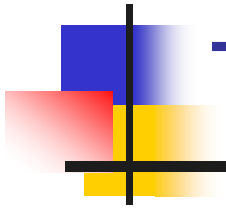
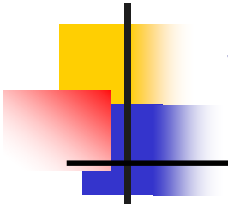


Technical System Structure: Technology or Enterprise?

Dan Johnsson

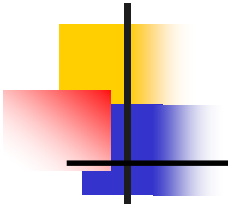
Omegapoint AB; Sweden

dan.johnsson@omegapoint.se



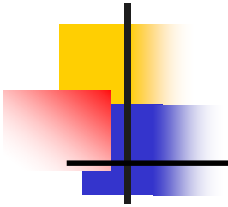
What We Do, Why We Care

- Going to build a system
  - or extend
- All requirements mined
  - Functional requirements
  - Non-Functional Requirements (NFR)



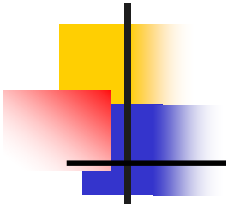
Inventory of Functionality

- Presentation
- Business
- Integration



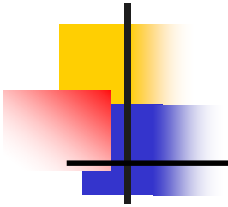
Deploy-Points

- Backend
- Central
- Peripheral



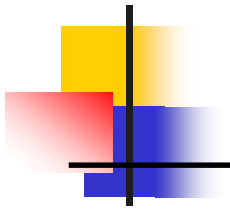
Structure

- What functionality in which deploy point?
- Presentation, Business, Integration
- Peripheral, Central, Backend



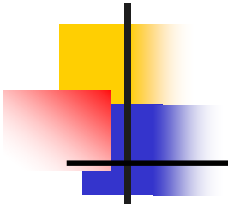
Capabilities

- Makes the difference
- Depend on system as whole
- Performance
- Capacity
- Reliability
- Security
- Extensibility



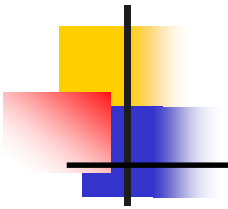
Capability Trade-off

- Cannot have all
- Architectual transformation
  - give up some of one
  - gain some of other
- Examples
  - Encryption (performance –> security)
  - Optimistic locking (reliability –> capacity)



Good or Bad?

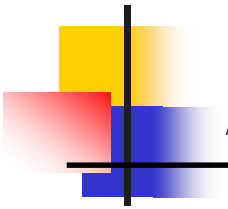
- Good if fulfils NFRs
- Else: apply transformations



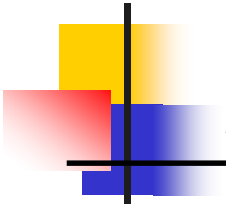
Origin of NFRs

- NFRs defined by business
- Eval driven by business risks
- Ebay *vs* Bank

- Architecture is Good if eliminates business risks

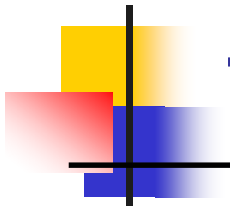


Anything Interesting Yet?



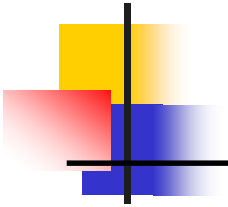
Architecture

- Technical structure of system
- *Structure functionality into components in such a way that all NFRs are fulfilled*
- Requires deep technical knowledge of component technologies

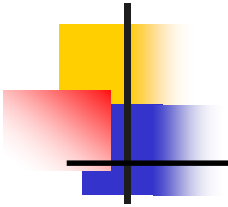


Two Models

- Architecture too big subject
- Restricted discussion
- Technology-based data-flow-architecture
- Enterprise-based abstraction-architectures



Dataflow Architectures

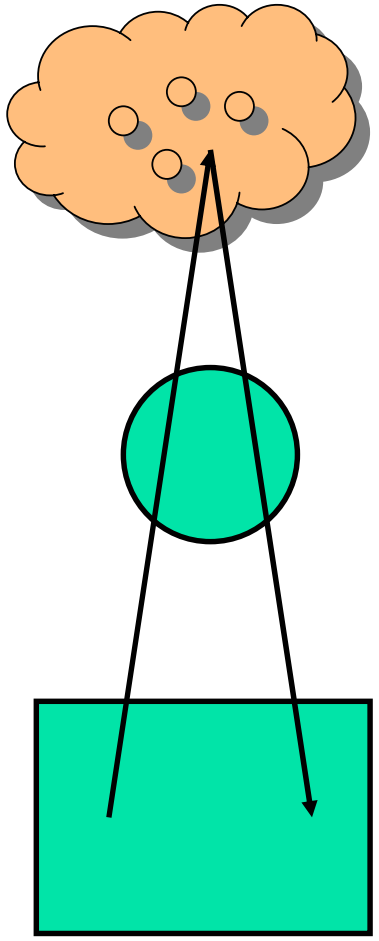


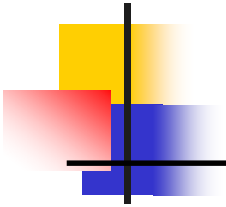
Based on Technology

- We build what we know
- Can be a good choice
  - Fast initial development
- Should be conscious
- Can be result of "design by coincidence"

Typical Realisation

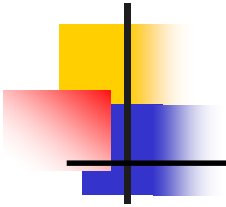
- Data pump in centre
  - *e.g.* Session EJB
  - designed ∅-state
- Architectural methods
  - few, load/store-style
- Argument data classes/structs
  - Fat in data
  - Poor in behaviour
- Logic in peripheral deploypoints





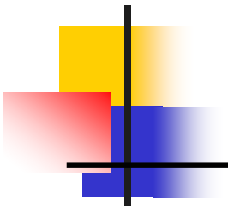
Example (Bank)

- Errand system
- Tuxedo backend
- EJB
  - fetchCustomer()
  - updateCustomer()
- Arguments: data graphs
  - CustomerVO, ContactVO, AccountVO
  - Getters/setters
    - Primitives
    - Relations
- GUI
  - Just one account of each type (saving, checking, *etc.*)



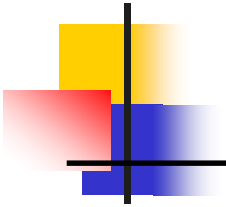
Evaluation

- Some scenarios
- Study impact



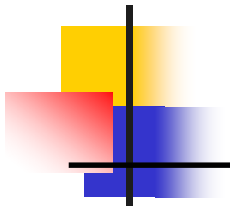
Scale Capacity

- Buy bigger server
- Cannot scale *one* aspect
  - *e.g.* increase in direct stock buy/sell
- Must scale up "everything"



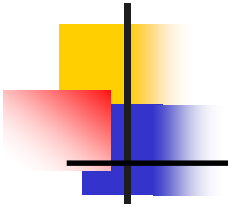
Business Transaction

- Multi step update
  - Incremental
  - Fast fail (*vs* system state "= reality")
- System has no concept of constraints
- No protection against inconsistent updates
- Have to pass all data in one call



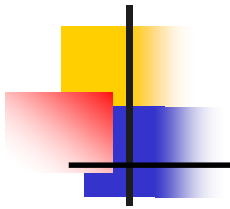
Non-Trivial Access Control

- Non-trivial
  - not just binary
  - *e.g.* role-based
- Example permissions
  - Cashier: change balance, not create account
  - On-line customer: move money
  - Telephone bank: create account
- System cannot tell them apart



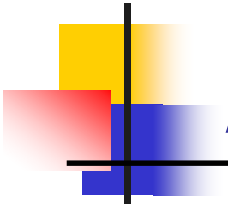
System Integration

- Business constraints enforced in client
- Business constraints enforced in client systems
- Mess



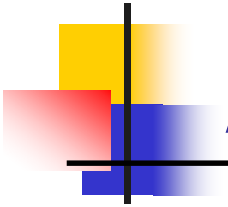
Summary of Capabilities

- Short-term extensibility +
- Scaling -
- Reliability -
- Security -
- Long-term extensibility -



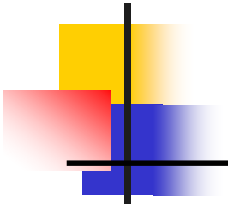
Abstraction Architectures

- All Computer Programmes are Simulations



Abstraction

- Abstract = part of organ
- Abstraction = simplification
- Hides technical realisation
- Model of conceptual understanding

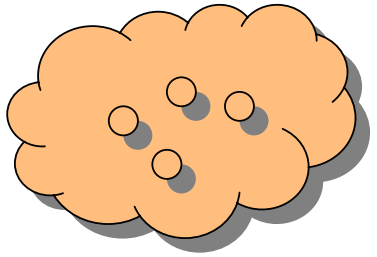


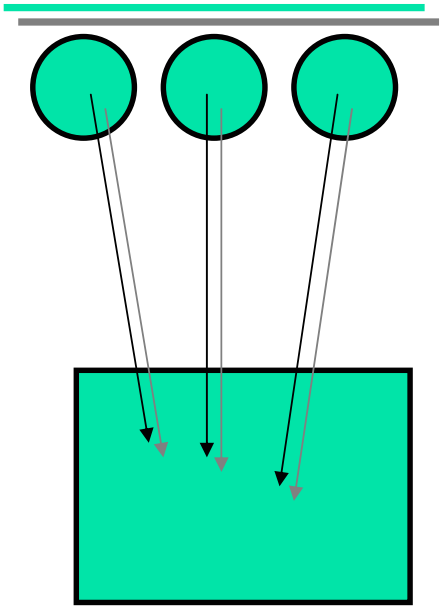
Based on Enterprise

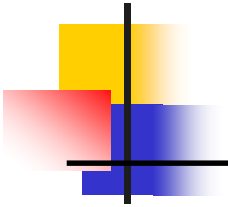
- Models understanding of enterprise
- Hard
  - Very hard
  - Budget for analysis & design
- Tips: Do not spend entire budget at once
  - Save some for later insights

Typical Realisation

- Service components in centre
  - Session EJB
  - State if appropriate
- Argument data object
  - Slim in data
  - Rich in behaviour
- Clear "system border"
- Logic in central deploy-points

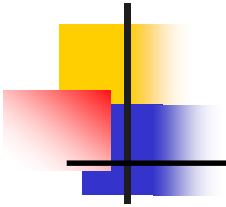






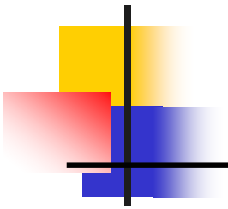
Example (Bank)

- Counter Bazaar
- Tuxedo backend
- Session EJBs:
  - AccountManagement
  - MoneyMoving
  - StockDepau
- Arguments: objects
  - Amount, ContactInfo



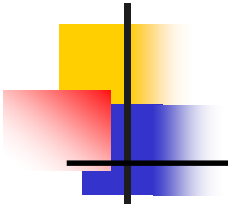
Evaluation

- Some scenarios
- Study impact



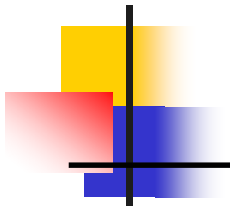
Scale Capacity

- Redeploy popular service
- Separate deploy / clustering
- Can scale up part



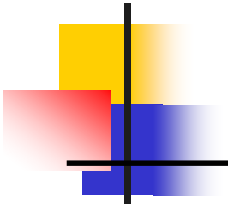
Business Transaction

- Constraints inside system
- EJB enforces consistent update
  - Can check *vs* backend
  - Fast fail
- Can span several calls
  - later "finalization"



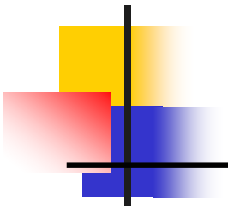
Non-Trivial Access Control

- Session methods match enterprise events
- Natural place to plug in access control
- Cashier: change balance, not create account
- On-line customer: move money
- Telephone bank: create account



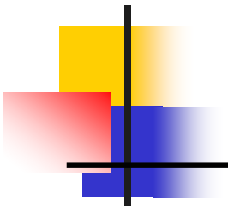
System Integration

- Business constraints enforced in system
- Client systems must build calls
- Easy to publish for integration
- (Web Services hype is about integration)

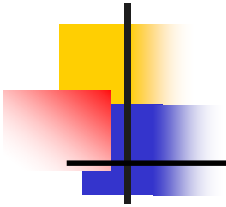


Summary of Capabilities

- Short-term extensibility -
- Scaling +
- Reliability +
- Security +
- Long-term extensibility +

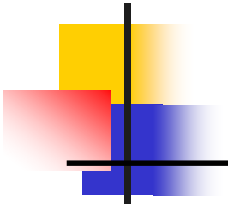


Getting It Done



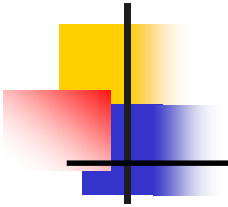
Myth of Doing It Right

- "Can't afford to re-implement"
  - Can afford to keep it?
- "We'll schedule a redesign later"
  - Not a show-stopper
- "Next system"
  - Trade-offs will be same
- "If it works, don't touch it"
  - "works"?



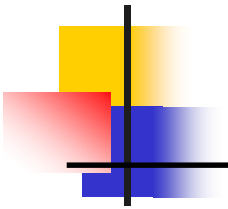
Migrating the Structure

- New Session EJBs publish business method
- Uses tech-oriented function
- Change client code one-by-one
- (Some time might pass)
- Hide tech-oriented function



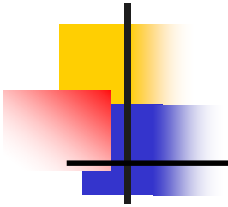
Problem En Route

- Long parameter lists
  - Lift business tier functionality
- Bloated Session EJBs
  - Splitting sessions



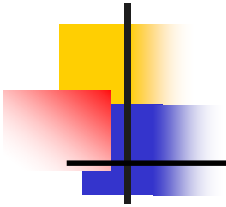
Lifting the Business Tier

- Get validation out of there
- Rich data objects
  - "business logic value object"
  - Date, CurrencyCode
  - SupplierCode, PhoneNumber
- Struts/JSF
  - actions create objects
  - Validation (form/validator)
    - Static Amount.wellformed(String)



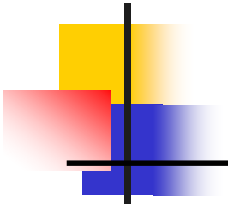
Splitting Sessions

- Doing-it-all-session
- Draw state diagram
- If state have several “aspects” – split
- Split
  - externally – two Session EJBs
  - internally – two internal states: same EJB



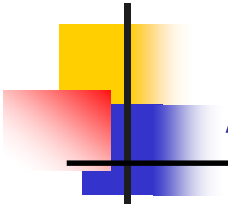
Trade-offs

- Early in system
  - Speed-of-development important
- Late in system
  - Other capabilities important
- Maturing systems/components
  - Trend
  - More enterprise based
  - Less tech based

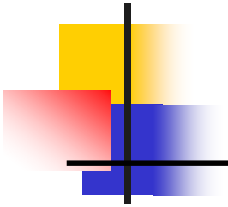


Conclusions

- Architecture is technology
- Tech architectures often fast
- Business architectures often better
- Can go from one to the other



Anything Interesting at All?



Stonecutters' Guild

- We, who cut mere stone, must always envision cathedrals