

Agile Architecture and Design

Pradyumn Sharma

pradyumn.sharma@pragatisoftware.com

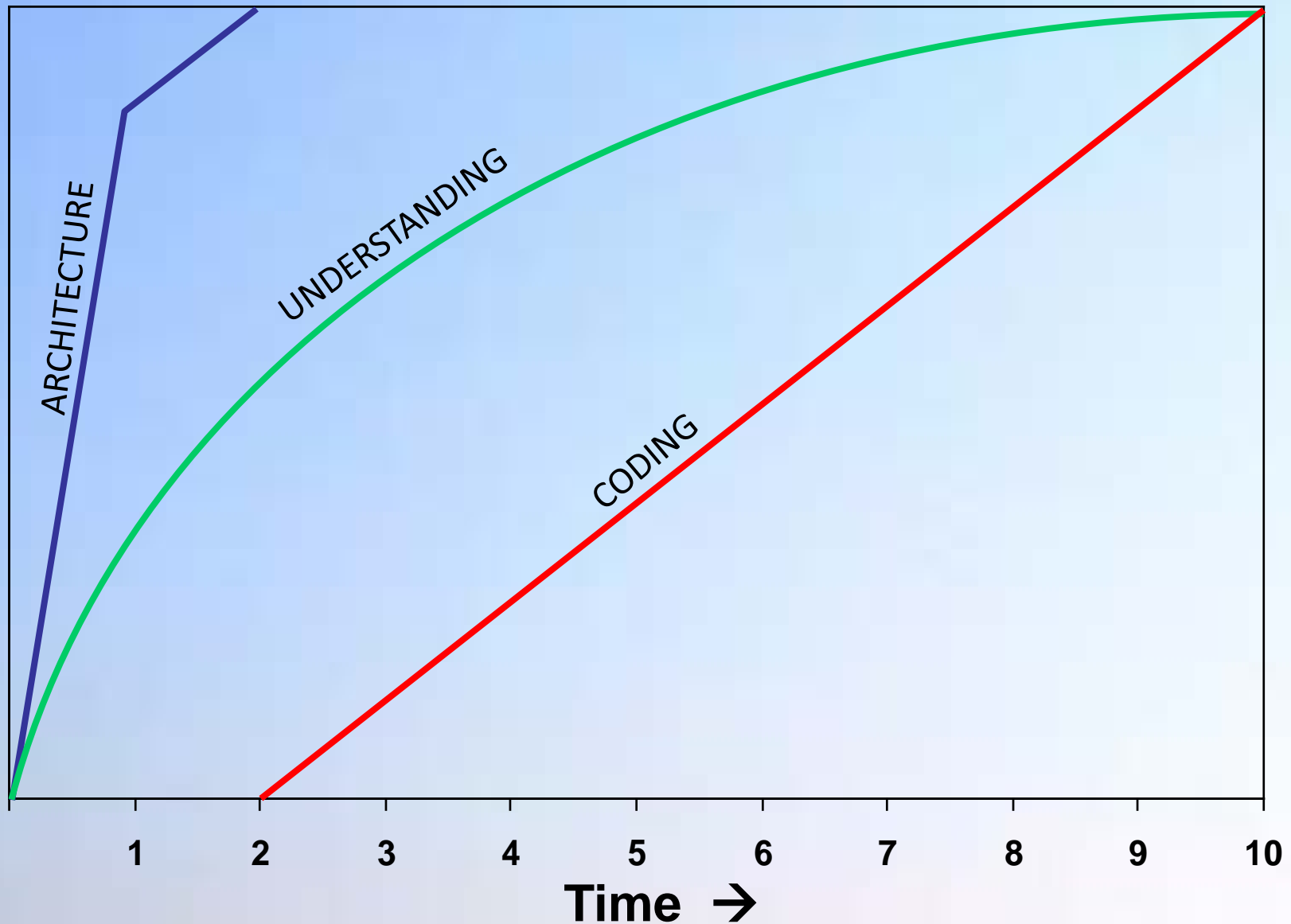
www.twitter.com/PradyumnSharma

Presenter: Pradyumn Sharma

- CEO, Pragati Software, Mumbai, India (www.pragatisoftware.com).
- About 30 years in the IT industry.
- Training and consulting: Agile methodologies, Solutions Architecture, Enterprise Architecture, NoSQL.



Software Architecture: When?





Agile Architecture and Design

- Evolve iteratively
- through
 - an initial envisioning
 - implementation of stories
 - refactoring and restructuring

Agile Architecture and Design: Overview

- Basics:
 1. Involve the entire team
 2. Have an “Architecture Owner” role
 3. Understand your product
- The key stuff:
 4. Create and architecture vision
 5. Build architecture through stories
 6. Model and implement incrementally

#1. Involve the Entire Team

Involve the Entire Team in

- Product Backlog Grooming
- Domain modeling
- Architecture discussion, evolution, implementation
- Architecture reviews
- Technical debt sessions
- Refactoring and restructuring

#2. Have an “Architecture Owner” Role

Responsibilities

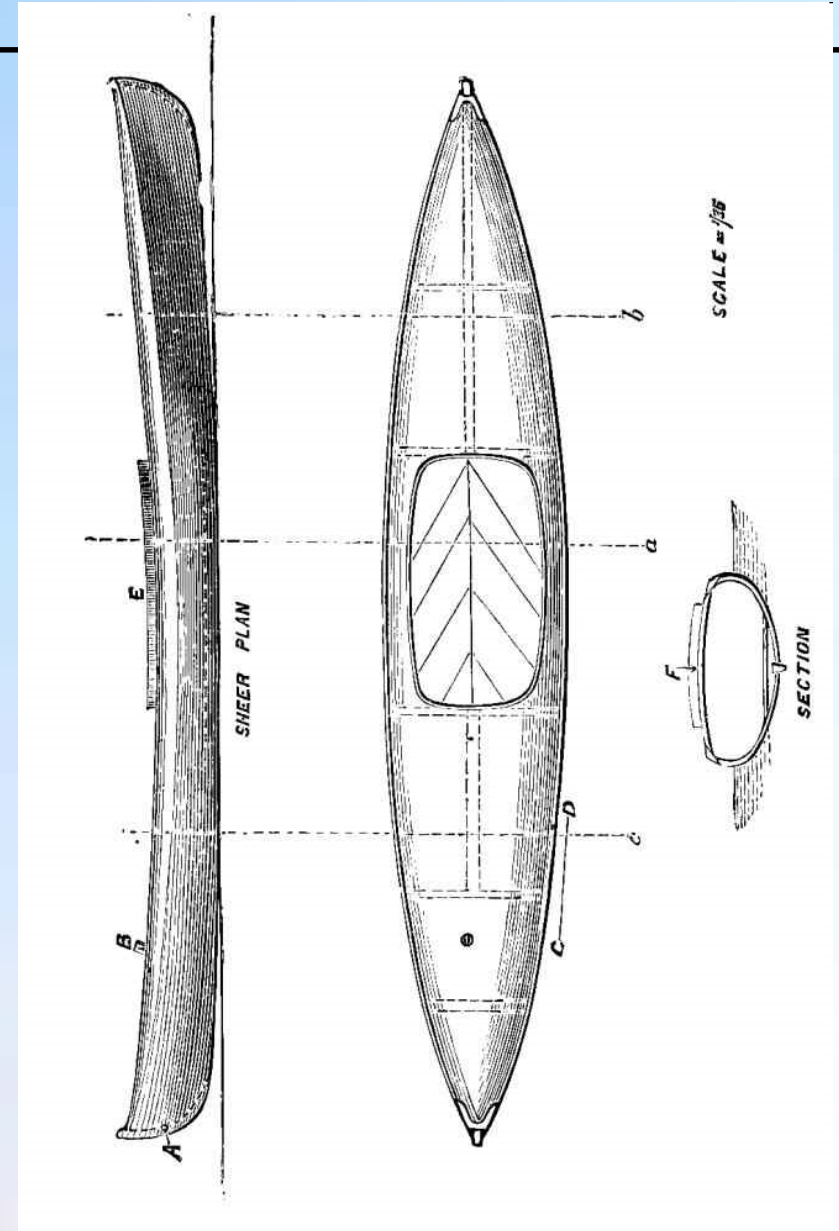
- DOs:
 - Bring the team together for all discussions regarding architecture envisioning and modeling
 - Facilitate architecture modeling and evolution
 - Help in building a shared understanding
 - Help the team members enhance their capabilities in understanding architectural principles and tradeoffs involved
- DONTs:
 - Dictate the architecture, preventing others from having their say.
 - Guard the architecture as personal property.

#3. Understand Your Product

Understand Your Product



Understand Your Product



#4. Create an Architecture Vision

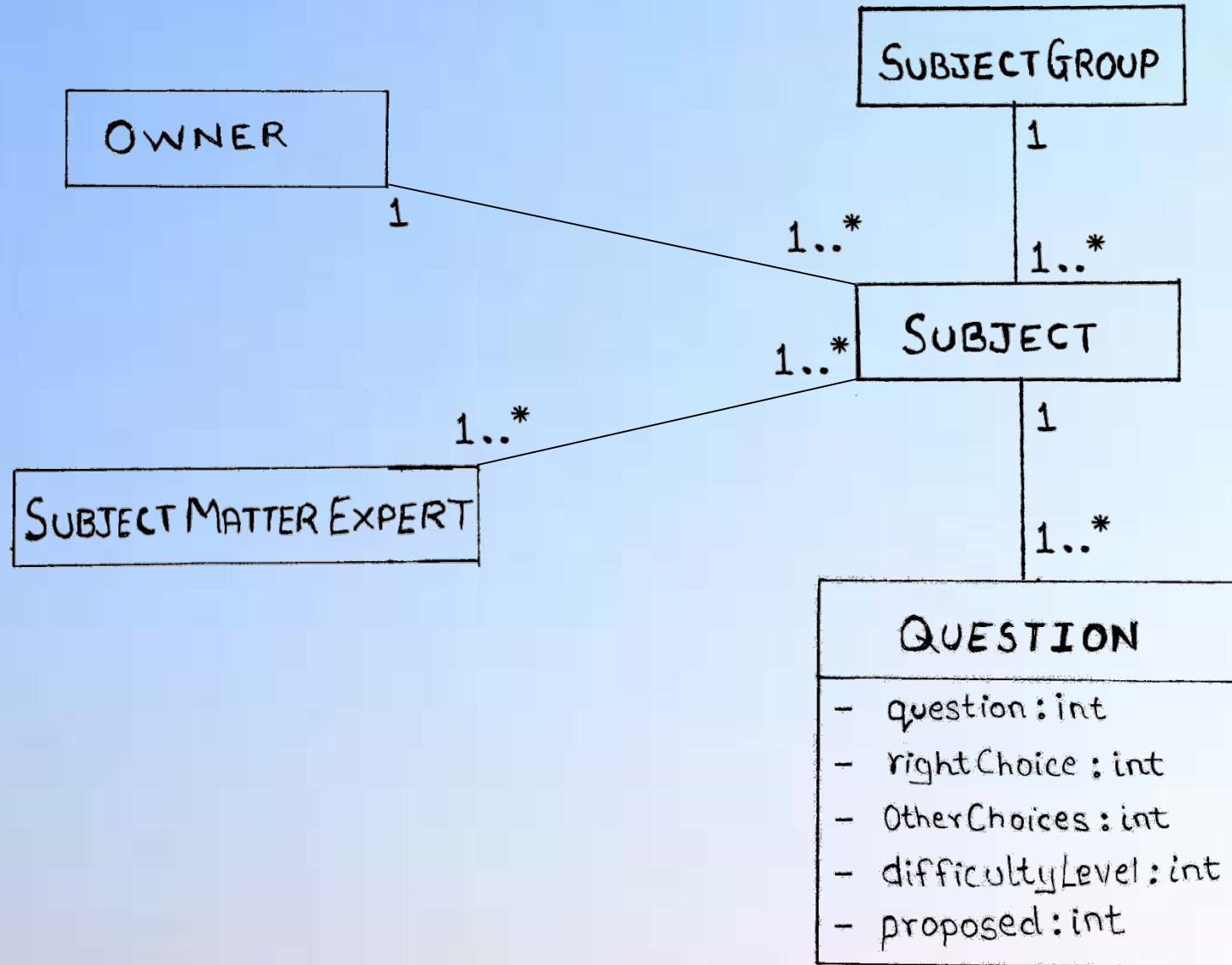
Create an Architecture Vision

- When?
 - Sprint zero.
- How?
 - Architecture / technical workshop.
- Who?
 - Team as well as the Product Owner (architecture must be based on requirements).

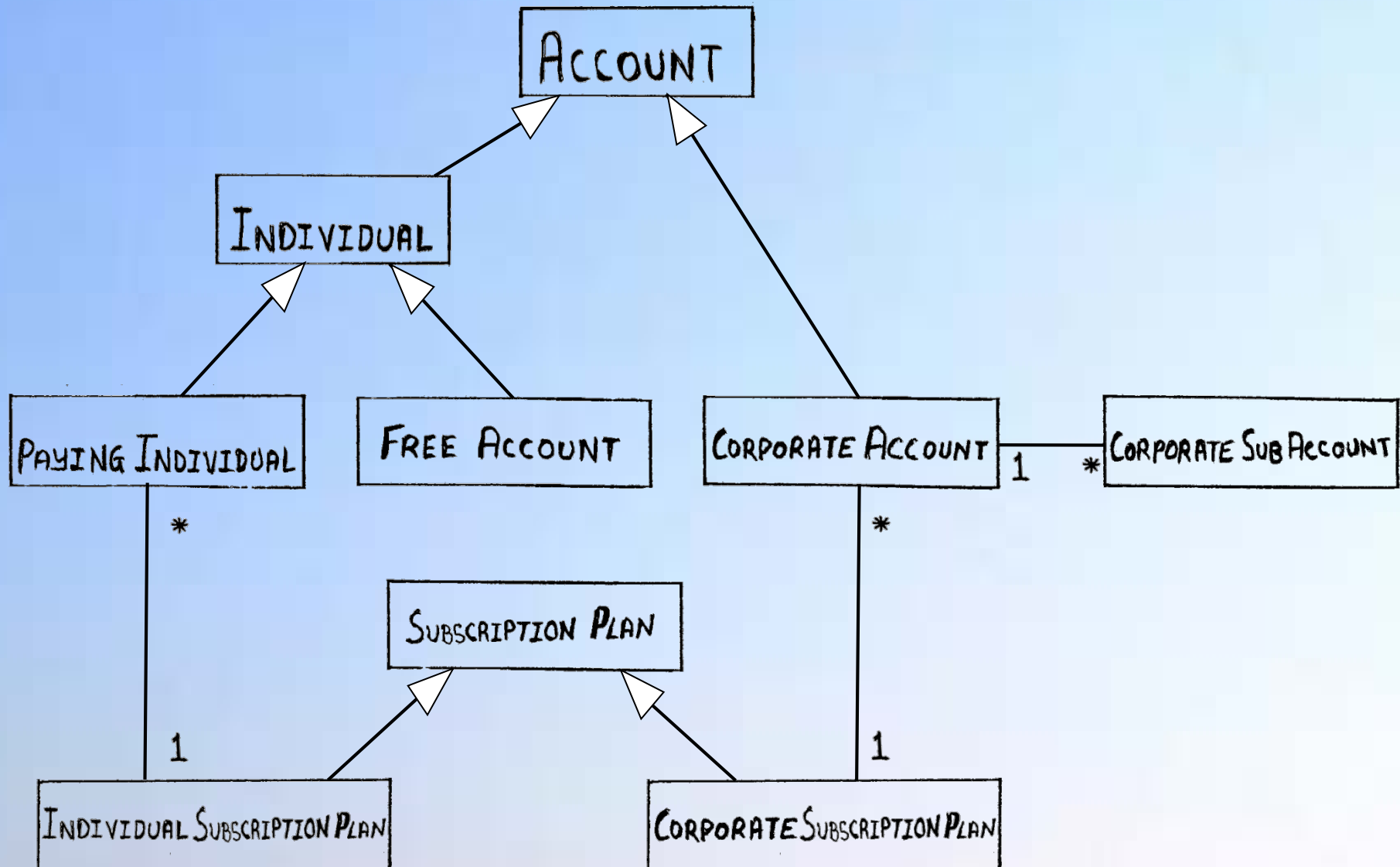
Architecture Workshop: Activities

- Domain modeling

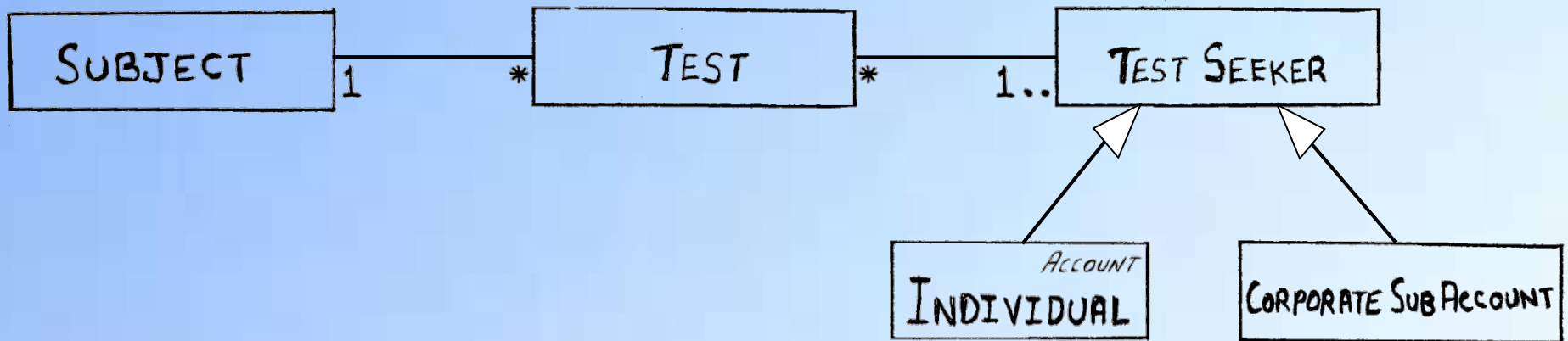
Domain Modeling



Domain Modeling



Domain Modeling



Architecture Workshop: Activities

- Domain modeling
- UI prototyping

UI Prototyping

ITEMS

☐ Category : _____ ↓ [More Filters]
 Sort by _____ ↓

Get data | Refresh

+ Add new Item

A B C X Y Z Others All

Showing 1-25 of ----
 << First
 < Prev.
 Next >
 Last >>

<input type="checkbox"/>	Item	Unit	Rate	Stock	----	
<input type="checkbox"/>	Alfa Black Toner Cartridge (AB001)	No.	\$ 56.00	---		Edit Delete
<input type="checkbox"/> ⚙️ ↓	AmeriPaper A4 size 500 sheets (AM001)	Pack	\$ 3.99	----		Edit Delete

UI Prototyping

Items > New Item

General Specifications Vendors

Item name _____

Code _____

Description

Unit _____ ↓

Save & Exit

Save & Continue

Cancel

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns

Architectural Qualities: Examples

- Portability
- Modifiability
- Performance
- Security
- Testability
- Usability
- Availability
- Conceptual integrity
- Accuracy
- Concurrency
- Customization points
- Internationalization
- Operations
- Maintenance
- Environmental impact
- Reliability
- Regulatory compliance
- Serviceability
- Support
- Dependencies on external systems

Architectural Qualities: Examples

- Portability
 - Modifiability
 - Performance
 - Security
 - Testability
 - Usability
 - Availability
- Conceptual integrity
 - Accuracy
 - Concurrency
 - Customization points
 - Internationalization
 - Operations
 - Maintenance
 - Environmental impact
 - Reliability
 - Regulatory compliance
 - Serviceability
 - Support
 - Dependencies on external systems

Search Engine:

- Performance
- Availability
- Portability

Stock Trading System:

- Security
- Dependencies on external systems
- Performance

Online Travel Agency:

- Dependencies on external systems
- Usability
- Internationalization

Portability Requirements: Examples

- System should run on all popular web browsers as well as smartphones.
- It should be easy to migrate data from one database platform to another.
- Customers should be able to choose the database platform for their respective deployments.

Security Requirements: Examples

- Preventing unauthorized access to data or services
- Dealing with DoS attacks
- Non-repudiation (a transaction cannot be denied by any party)
- Secure transmission of sensitive data

Usability Requirements: Examples

- Ability to save incomplete data as draft and resume later
- Appropriate feedback to the users about state of completeness of a business process / long-running transaction / multi-step data entry, etc.
- Ability to bookmark, roll back, etc.

Availability Requirements: Examples

- System uptime should be at least 99.9%
- When a system service fails, an alert to the Infrastructure team should be raised.
- Even when one service of the system becomes unavailable the other services should continue to run as far as possible.
- It should be possible to deploy newer versions of the components without shutting down the system.

Integrity Requirements: Examples

- System should detect any issues with data consistency or integrity that creep in.
- Likelihood of concurrent access to data: very high / medium / low. => Need for concurrent edits without compromising data integrity.

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Discuss how to achieve various desired qualities, but don't commit

Portability: Strategies

- Separation of UI logic, business logic, data access logic in separate components
- Layering, partitioning

Modifiability: Strategies

- Loose coupling between components
- Proper encapsulation
- Layering, partitioning
- Model-View-Controller
- Application framework providing common features and services, extension points for providing story-specific behaviour
- OO principles and design patterns
- Anticipating changes, providing hooks to facilitate changing behaviour
- Non-architectural aspects, such as coding conventions and techniques

Concurrency: Strategies

- Use of Transaction Managers
- Optimistic locking / Pessimistic locking.
- Fine-grained locking / Coarse-grained locking

Usability: Strategies

- Separation of the UI from the rest of the application
- Giving feedback about what the system is doing
- Letting the user issue commands such as Save as Draft, Cancel, Undo, show multiple views
- Design patterns: Command, Memento
- Maintaining a model of the task, or the system, or the user

Performance: Strategies

- Infrastructure planning: server capacity, clustering, failover, virtualization, network bandwidth
- Managing event rate
- Quantum of communication among components, layering
- Database: indexes, partitions, stored procedures
- Non-architectural aspects:
 - choice of algorithms
 - implementation of selected algorithms
 - writing efficient database queries

Security: Strategies

- Authentication of users
- Single sign-on
- Authorization of users, limiting access
- Audit trail
- Intrusion detection system

Integrity: Strategies

- Periodic run of batch programs to check integrity of derived data against raw data
- Discover patterns, fix integrity problems automatically, maintain history of such changes

Strategies for Achieving Architecture Qualities

- Discuss and note down how to achieve the various desired architecture qualities, but don't commit to them yet.

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Discuss how to achieve various desired qualities, but don't commit
- Identify cross-cutting requirements

Cross-Cutting Requirements: Examples

- Audit trail
- Alerts for important events that need attention
- Centralized error logging
- Excel export from all browse windows

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Discuss how to achieve various desired qualities, but don't commit
- Identify cross-cutting requirements
- Identify other considerations for architecture

Other Considerations: Examples

- Projected lifetime of the system
- Cost and benefit
- Time to market
- Rollout schedule
- Target market
- Correctness and completeness

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Discuss how to achieve various desired qualities, but don't commit
- Identify cross-cutting requirements
- Identify other considerations for architecture
- At every stage, validate with the product owner / customer / end users

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Identify cross-cutting requirements
- Identify other considerations for architecture
- At every stage, validate with the product owner / customer / end users
- Discuss how to achieve various desired qualities, but don't commit
- Identify potential technical risks

Potential Technical Risks: Examples

- Technology being used is not mature
- Product is very complex to implement
- Integration requirements are very complex and diverse
- Requirements are unclear, unstable

Killing the project minimizes risk, but also eliminates reward.

–James McGroddy, former CTO at IBM

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Identify cross-cutting requirements
- Identify other considerations for architecture
- At every stage, validate with the product owner / customer / end users
- Discuss how to achieve various desired qualities, but don't commit
- Identify potential technical risks
- Take stock of existing reusable architectural assets.

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Identify cross-cutting requirements
- Identify other considerations for architecture
- At every stage, validate with the product owner / customer / end users
- Discuss how to achieve various desired qualities, but don't commit
- Identify potential technical risks
- Take stock of existing reusable architectural assets.
- Prioritize the architecture features. Based on:
 - Business value
 - Cost of implementing early vs cost of implementing late

Architectural Features' Prioritization: Example

- Separation of UI, database, business logic, workflow logic into separate, distributed layers. (Benefits: portability, modifiability).
- Creation of an application framework. (Benefits: avoiding duplication of code in various stories; consistency of behavior; ease of adding cross-cutting requirements).
- Pessimistic Locking for concurrency management.
- Usability features in the framework: Save as Draft, Undo, Redo.
- Cross-cutting requirement: Audit trail.

Architecture Workshop: Activities

- Domain modeling
- UI prototyping
- Identify desired architecture qualities and concerns
- Identify cross-cutting requirements
- Identify other considerations for architecture
- At every stage, validate with the product owner / customer / end users
- Discuss how to achieve various desired qualities, but don't commit
- Identify potential technical risks
- Take stock of existing reusable architectural assets.
- Prioritize the architecture features. Based on:
 - Business value
 - Cost of implementing early vs cost of implementing late
- Include desired architectural qualities / strategies as product backlog items.

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

#5. Build Architecture Through Stories

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

Story 1: Creating List of Items

Acceptance Test

- ✓ 1.
- ✓ 2.
- ✓ 3.
- ✓ 4.
- ✓ 5.
- ✓ 6.

Implement the UI and the Story

ITEMS

☐ Category: [\[More Filters\]](#) Sort by

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#) [Others](#) [All](#)

Showing 1-25 of 497 [<<First](#) [<Prev](#) [Next>](#) [Last>>](#)

<input type="checkbox"/>	Items	Unit	Rate	Stock	
<input type="checkbox"/>	Alfa Black Toner Cartridge (AB001)	No.	\$56.00	Edit Delete
<input type="checkbox"/>	AmeriPaper A4 size 500 sheets (AM001)	Pack	\$3.99	Edit Delete

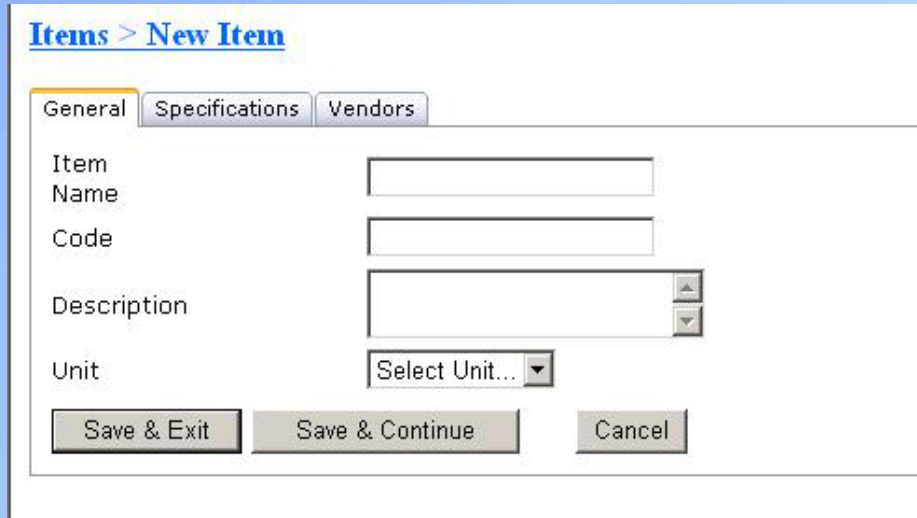
Form_load:

- fetch data from Items table
- populate the grid

New_click:

- display blank entry form

Implement the UI and the Story



The screenshot shows a web application interface for adding a new item. At the top, there is a breadcrumb trail 'Items > New Item'. Below it, there are three tabs: 'General' (which is selected), 'Specifications', and 'Vendors'. The 'General' tab contains the following fields:

- 'Item Name' with a text input field.
- 'Code' with a text input field.
- 'Description' with a text area (indicated by a vertical scrollbar on the right).
- 'Unit' with a dropdown menu showing 'Select Unit...'.

At the bottom of the form, there are three buttons: 'Save & Exit', 'Save & Continue', and 'Cancel'.

SaveAndExit_click:
validate data
INSERT into Items values ...
exit form
etc.

Implement the UI and the Story

ITEMS

☐ Category: [\[More Filters\]](#) Sort by

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#) [Others](#) [All](#)

Showing 1-25 of 497 [<<First](#) [<Prev](#) [Next>](#) [Last>>](#)

<input type="checkbox"/>	Items	Unit	Rate	Stock	
<input type="checkbox"/>	Alfa Black Toner Cartridge (AB001)	No.	\$56.00	Edit Delete
<input type="checkbox"/>	AmeriPaper A4 size 500 sheets (AM001)	Pack	\$3.99	Edit Delete

Form_load:

- fetch data from Items table
- populate the grid

New_click:

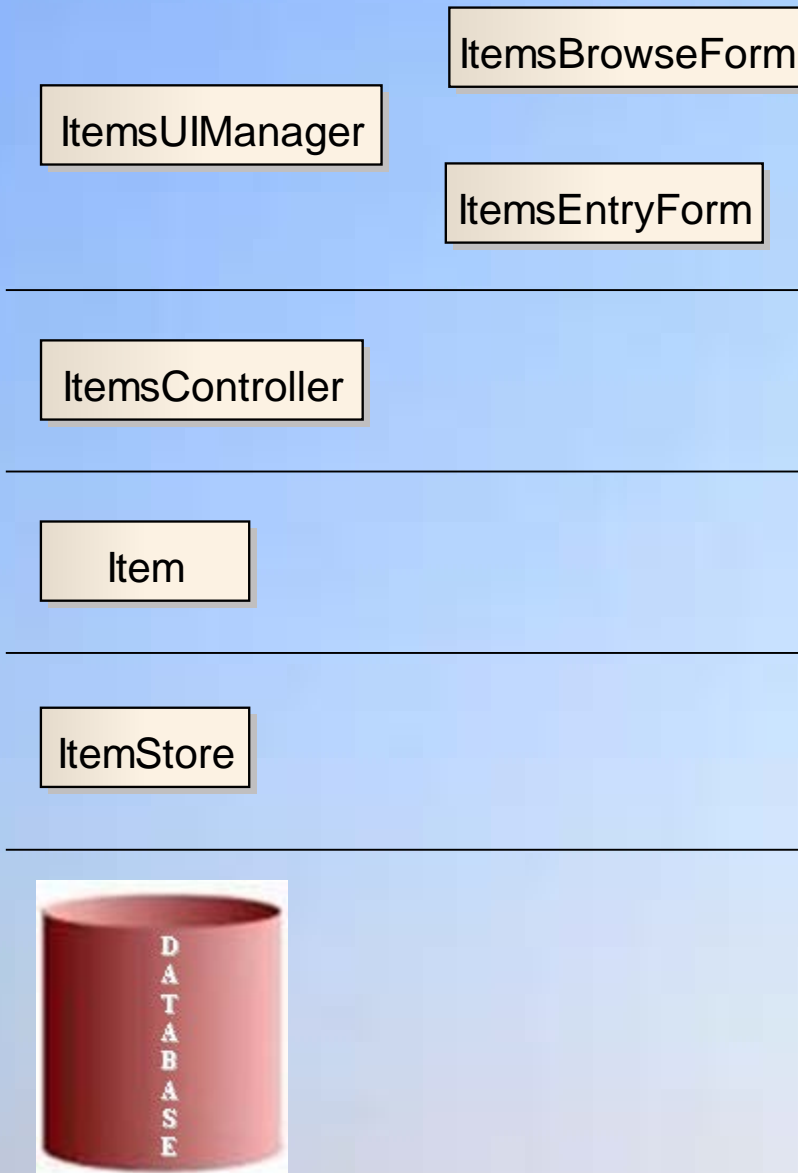
- display blank entry form
- //...on return from the entry form
- re-populate the grid from Items table

etc.

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

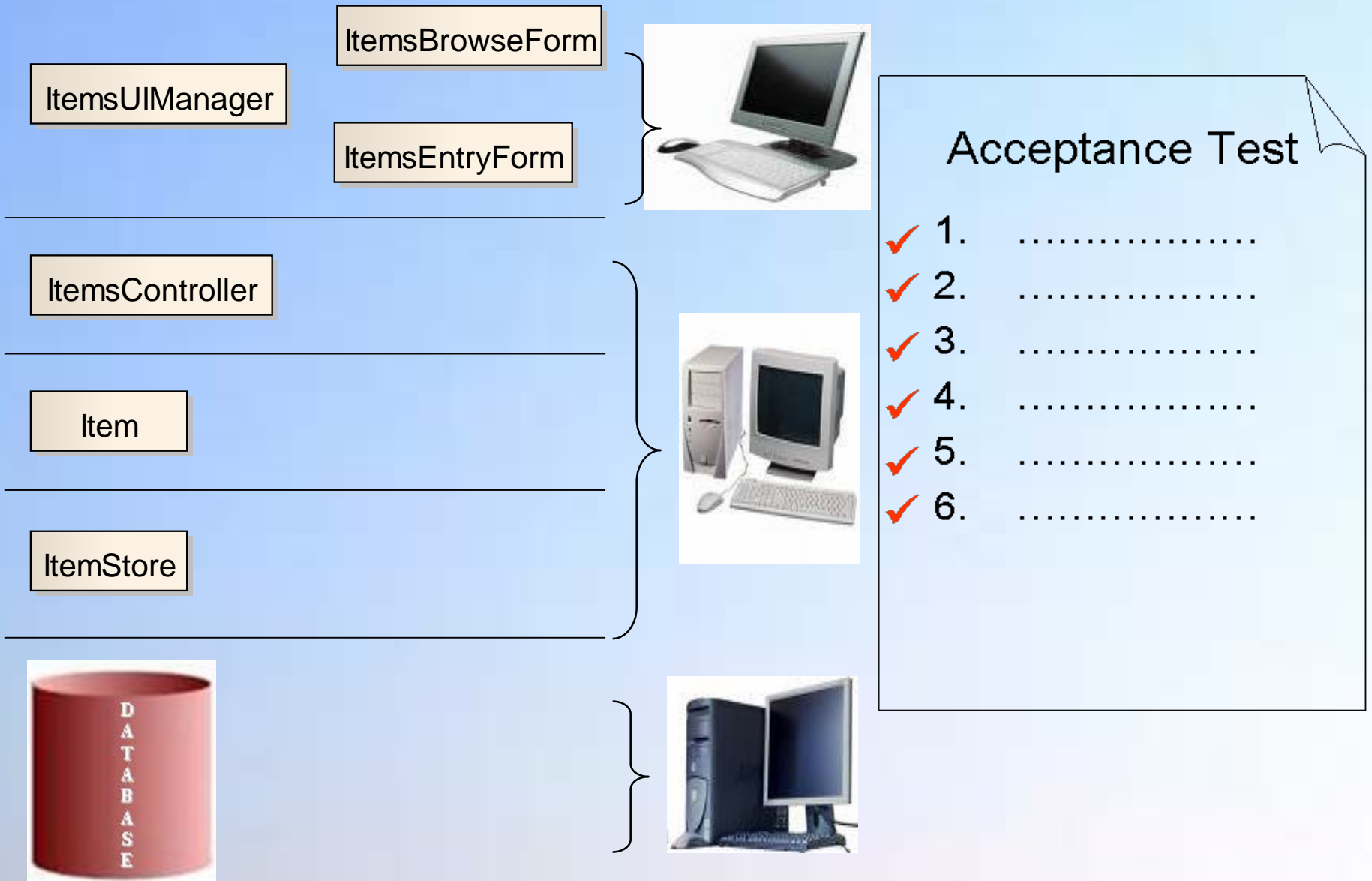
Story 2: Layered Architecture



Acceptance Test

- ✓ 1.
- ✓ 2.
- ✓ 3.
- ✓ 4.
- ✓ 5.
- ✓ 6.

Distributed Architecture



Looking Back

- Miniscule architecture evolved, proven with the code
- Architectural stability and state of completeness: very low

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

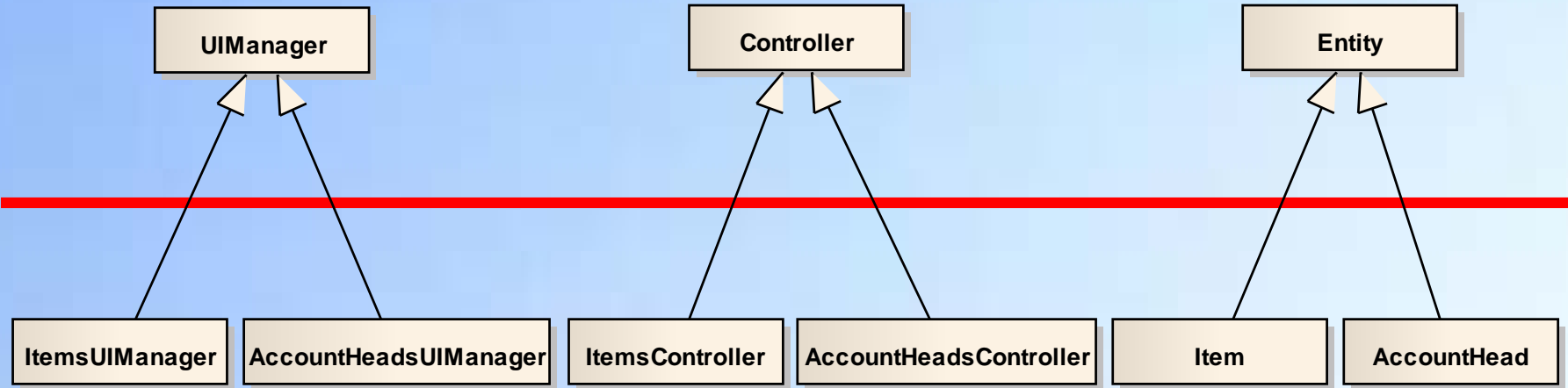
Story 3: Creating List of "Account Heads"

- Write acceptance tests.
- Design the UI.
- Start implementing the story.
- Similarity in implementation with the earlier story...
=> take up the next story also

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

Story 4: Application Framework Evolution



Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

Story 5: Test Framework, "Tax Codes" Story

- Implement a third story, by extending the framework.
- Helps in smoothening some rough edges in the framework.

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

Story 6: Framework, Handling Concurrency

- Implementing database record locking in the framework, instead of in individual stories.
- Locking strategies:
 - Pessimistic locking
 - Optimistic locking
 - No locking
- Who chooses the locking strategy?
- And how do they choose it?
- How do we help them make a choice?
- By writing scenarios with UI prototyping.

Product Backlog

Product Backlog	
Sr.No.	Story
1	Maintaining the list of "Items" for stock and sale
2	Implementing Layering and Partitioning in "Items" story
3	Creating the list of "Account Heads"
4	Refactoring the "Items" and "Account Heads" stories to extract common behaviour into a "framework" for the application
5	Testing the framework by extending it to implement a third story (Tax Codes), and ironing out issues.
6	Implementing concurrency handling in the framework and testing
7	Account Heads: for "Customer" types, entering additional details
8	Framework: Including support for Save as Draft, Undo, Redo.
9	Framework: Audit Trail.

Similarly...

- Story 7: Account Heads, entering additional details for "Customer" type
- Story 8: Framework, support for Save as Draft, Undo, Redo
- Story 9: Framework, Audit Trail
- Likewise for other cross cutting functional or UI requirements. Examples:
 - Change history
 - Pagination in browse windows
 - Search within a browse window
 - Excel export from all browse windows

#6. Model and Implement Incrementally

Model and Implement Incrementally

- Model throughout the lifecycle, in small increments.
- Split large, complex stories.
- Example: Trainer tracking chart in a Training Monitoring System.

Example: Trainer Tracking Chart

<div> February , 2008 Previous Next Refresh Export Confirm W/O New Confirm Edit Edit History Exit </div>														
Trainer Tracking Chart For : February, 2008 [Total Trainings = 269.25 (Internal) + 100.75 (Freelancer) = 370 day(s)]														
Trainer(s)	9-Sat	10-Sun	11-Mon	12-Tue	13-Wed	14-Thu	15-Fri	16-Sat	17-Sun	18-Mon	19-Tue	20-Wed	21-Thu	22-Fri
Vipul	Citigro									#Intelen				#Heinz In
Foram	Tata S		+Relianc	+ [2] Kirlo	+ [2] Kirlo	Leave.	+Tata S	+Tata S			Leave .	+Reliance	+Reliance	Leave.
Varsha	Relianc			#Iflex So	#Iflex So	#Iflex So	#Iflex So			+Relianc	+Relianc		+Lehma	+Lehma
Viraj	Godrej			#Abbott I	#Abbott I		#Accel F	#Bharti			#Transoc	#Clariant	#Clariant	+Dun &
Swati	[2] Reli		+Mahind	Leave	Leave		#Nelco L	#Nelco L				Leave		
Anuradha	Dewan		#The Gre	#The Gre	Leave.	Attendin	Attendin		#Bank of			+MRC Tra		
Mrinal			Attendin							AttendTr	AttendTr			
Rajesh	Larsen	Leave.		Leave.	Prep On	Prep On	+Bharti							+Mazag
Altaf	S G Ve	\$S G Ve			\$Syntel I	\$Syntel I	\$Syntel I			+ICICI Pru	+ICICI Pru	+The Cre		+ICICI Pru
Manisha			# [2] Lehn	# [2] Lehn	#Panora	+ [2] Leh	+ [2] Leh	#khaitan		#Iflex So	#Iflex So	#Iflex So	#Panora	
Zubair	Internal t			Travel	\$Godrej	\$Godrej	Travel			#Transoc			+Godrej	+Godrej
RajVaidya	AFL Pri		Leave	Leave	Leave	Leave	Leave	Leave		Leave.		#Transoc	#Transoc	#Transoc
Amardeep	Videsh		+Siemen	+Siemen			#Merck S	#Merck S		#Hindust	#Hindust	#Hindust	+ [2] Tata	+ [2] Tata
SujeetK	Mercat		#Hindust	#Hindust	#Hindust	Leave	Travel.	\$ICICI P	\$ICICI P	Travel.	+ [2] Pru	+ [2] Pru	+Viacom	
Chandra	eave.		Evaluato	Evaluato	+ [2] Leh	+ [2] Leh	+ [2] Leh	+ [2] Leh		Attend "	Attendin	Attendin	Attendin	Attendin
Bhavna	eave.		#Datam	#Datam	#Datam	#Datam	#Datam	#Datam		#Datam	#Datam	#Datam	#Datam	#Datam
Shraddha			Cert On	Attendin						+Bave T	+Bave T	+Bave T	+Bave T	+Rave T
Avnika	Half Day		Leave.											
Abhishek			Leave.	Attendin	Attend L	Attend L								+ [2] Leh
Anushka				Cert On	Attend L	Attend L								Attend T
Ankur			Part for											
Vishal	Larsen	\$Larsen	\$Larsen	Unavaila	Unavaila	Unavaila								Unavaila
Merlin	Larsen		#Larsen	#Larsen		Prep For								Half Day
Minal	eave.	Leave.	Leave.	Attendin	Leave.									Attend T
Aruna		+ [2] Gau												
Pradyumn														
Majrul	DC Off			\$Mastek	\$Mastek	#Mastek	#Mastek			#Mphasis	#Mphasis	#Iflex So	#Iflex So	#Iflex So
Yatin Naik														
Chetan														
Ganesh B			\$Capge	\$Capge	\$Capge									
Girish Rao	Larsen		#Larsen	#Larsen			#Larsen &	#Larsen &		#Larsen &	#Larsen &	#Larsen &	#Larsen &	
Chitra	Rave T							+Rave T	+Mahes				+CFC In	+CFC In
Vaishali										+Lodha	+Lodha	+Lodha		

Work order Number C/2007/2095 - S
Client location MIDC , Knowledge Center
Location Datamatics Limited - 15
Subject Visual Basic .NET 2.0
Timings 09:30 AM to 05:30 PM
M E Rosy
Expected Pax.: 15 Actual Pax.: 12

Day 1 of 3

Model and Implement Incrementally

- Static grid with hard-coded data. Simple navigation keys.
- Dynamic grid, data retrieval from database.
- Cache table. Auto-refresh. Comments in cells.
- Navigation across months.
- Cut, Copy, Paste, Excel export options.
- Options to add / edit work orders from the grid.
- Tracking of incremental changes. Change history.

Static grid with hard-coded data. Simple navigation keys.

Dynamic grid, data retrieval from database.

Cache table. Auto-refresh. Comments in cells.

Navigation across months.

Cut, Copy, Paste, Excel export options.

Options to add / edit work orders from the grid.

Tracking of incremental changes. Change history.

Model and Implement Incrementally

- At the start of each iteration, during the sprint planning meeting, have discussions on incremental modeling, design changes.
- Apply architecture and design patterns as required, gently.
- Don't worry about getting your architecture right on the first day.
- Test-driven development.
- Refactoring.
- Build automation.
- Continuous integration.

Questions?

Thank You

Pradyumn Sharma

pradyumn.sharma@pragatisoftware.com

Twitter: PradyumnSharma