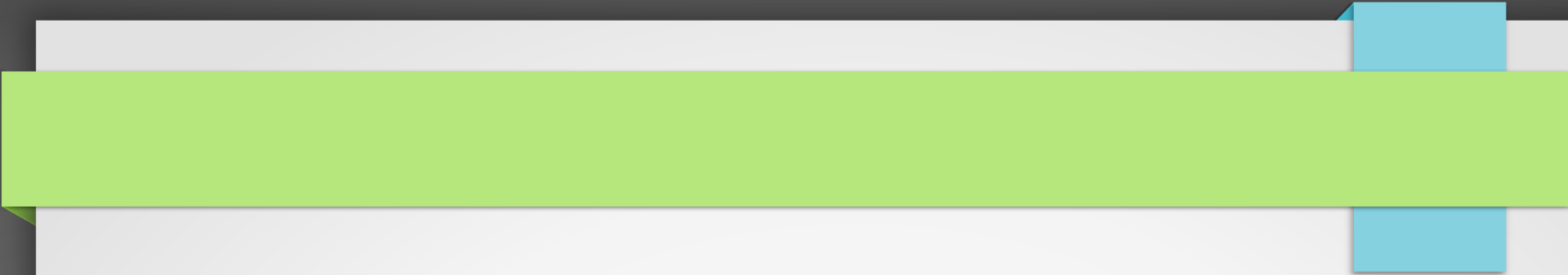


# Enabling Agility Through Architecture

1201310026(Aayush Kumar Singha)

1201310031(Aryan Srivastva)

1201310049(Devaraj Phukan)



“Get me an 80% solution NOW rather than a 100% solution two years from now and help me innovate in the field”

- Z. Lemnios, Director of Defense Research and Engineering, USA.

# Introduction

- Industry and government stakeholders continue to demand increasingly rapid innovation and the ability to adjust products and systems to emerging needs.
- Amongst all the enthusiasm for using Agile practices to meet these needs, the critical role of the underlying architecture is often overlooked.

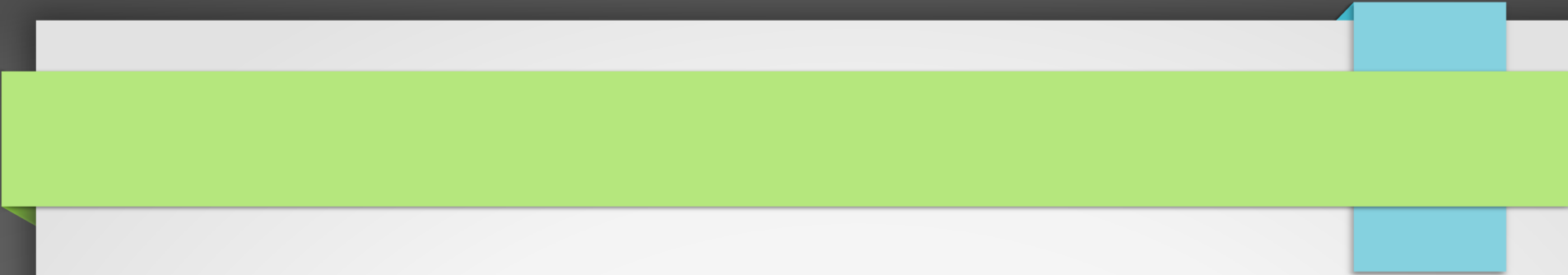
# Agility

- Ability to change, grow, adapt, enhance, increment, evolve .....
- **Enhancement agility** : Improvement/Increment in the features.
- **Architectural agility** : Ability to accomodate the enhancements in the architecture of the system.

# The waterfall model comparision.

Under the Waterfall paradigm of software development, an extensive **requirements phase** is conducted to anticipate needs for both the initial and subsequent releases of the product or system being developed.

Once the architecture is implemented, **Enhancement Agility** can be achieved, provided that the emergent user needs **fit within the boundaries anticipated** during the requirements phase.



In contrast to Waterfall methodologies, Agile software development focus on delivering observable benefits to the end users through **working software, early and often.**

A backlog of functional **“user stories”** is created. These stories are **prioritized** by end users and/or the product owner.

Development teams draw stories from the backlog and implement them in **accordance with end-user prioritization scheme.**

The Agile community's focus on **continuous delivery of user-valued stories** is **another means of achieving Enhancement Agility.** However, this approach also has its shortfalls, stemming mainly from an **inadequate focus on dependency analysis.**

## Problem with prioritization scheme

- Individual stories cannot be regarded in isolation. Stories have **dependencies on other stories**.
- Similarly, stories have dependencies upon the **architectural elements** of the system.
- These dependencies exist regardless of **domain stability or technical maturity**. They exist regardless of whether the system is in its initial development stages or has been deployed and has been in the field for several years.

# Architectural Agility

Architectural Agility addresses shortcomings that occur within both the Waterfall and the Agile lifecycle models.

Architectural Agility maintains a steady and consistent focus on continuing **architectural evolution** in support of emerging customer-facing features.

The architecture should not **over-anticipate** emergent needs, delaying delivery of user value and risking development of overly complex and unneeded architectural constructs. At the same time, it should not **under-anticipate** future needs, risking feature development in the absence of architectural guidance and support.



## Enhancement + Architectural Agility

Figure 1: Agile iteration planning – focus on User Stories

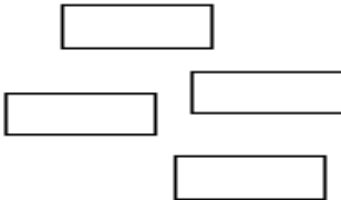
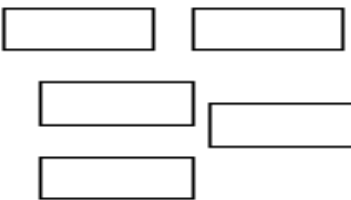
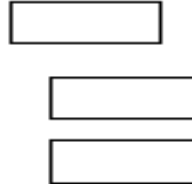
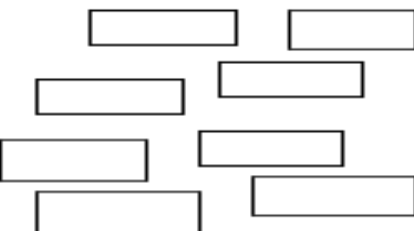
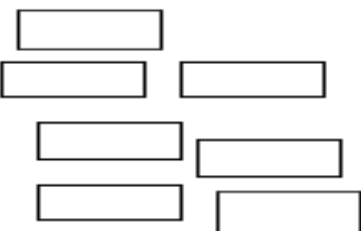
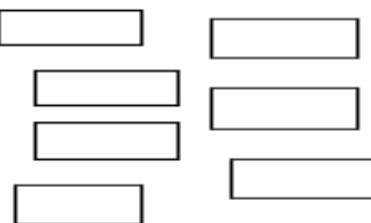
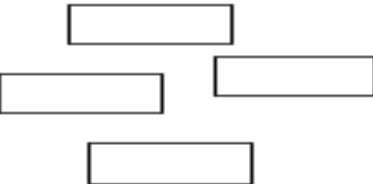

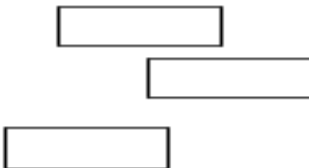
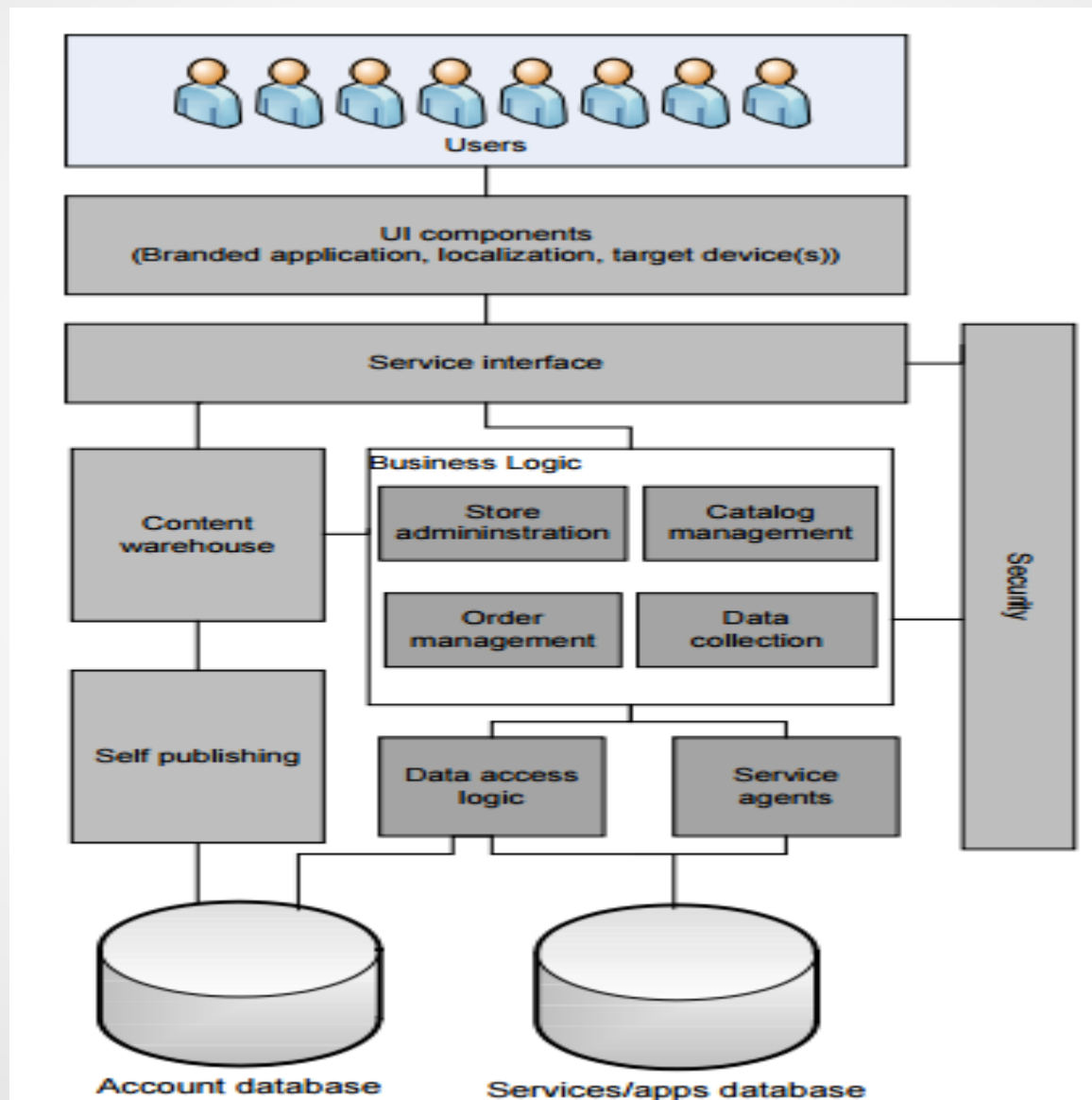
	Iteration 1	Iteration 2	Iteration 3
User Stories			

Figure 2: Architectural elements in agile iteration planning

	Iteration 1	Iteration 2	Iteration 3
Capabilities			
Architectural Elements & Technical Research			

# Conceptual App Store Architecture



# Dependency Structure Matrix

		Reporting	App catalog management	Sales management	Upgrade management	Promotion management	Order management	Account management	
Capabilities	Reporting		x	x	x	x	x	x	6
	App catalog management	x		x	x	x	x	x	6
	Sales management		x			x	x	x	4
	Upgrade management		x						1
	Promotion management	x	x	x			x	x	5
	Order management	x							1
	Account management						x		1
Architectural elements	UI components	x	x	x	x	x	x	x	7
	Service interface	x	x	x	x	x	x	x	7
	Content warehouse	x	x		x				3
	Self publishing	x	x		x				3
	Account database	x	x				x	x	4
	Services/apps database		x		x				2
	Data access logic		x				x	x	3
	Service agents		x		x	x	x	x	5
	Security		x	x	x	x	x	x	6
	Order management	x					x	x	3
	Data collection	x						x	2
	Store administration	x	x	x	x	x	x	x	7
	Catalog management	x	x		x				3
		9	11	4	9	5	8	9	

# Architecture Heuristics Focussed on Value

- Real Option Analysis
- Technical Debt Analysis
- Real options analysis and technical debt management offer potential models to make an informed choice and find the right balance of agility, innovation, and speed on the one hand, and governance, flexibility, and planning for future needs on the other.

# Real Option Analysis

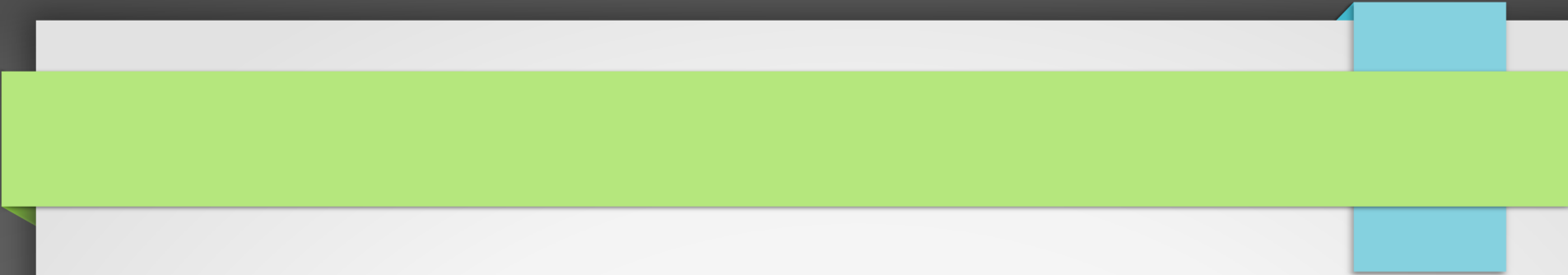
- It is a financial analysis model
- Helps to determine
  - How much upfront cost should be spent?
- Quantify value of flexibility into real assets
- Business decisions to determine value of delayed decision making

# Technical Debt Analysis

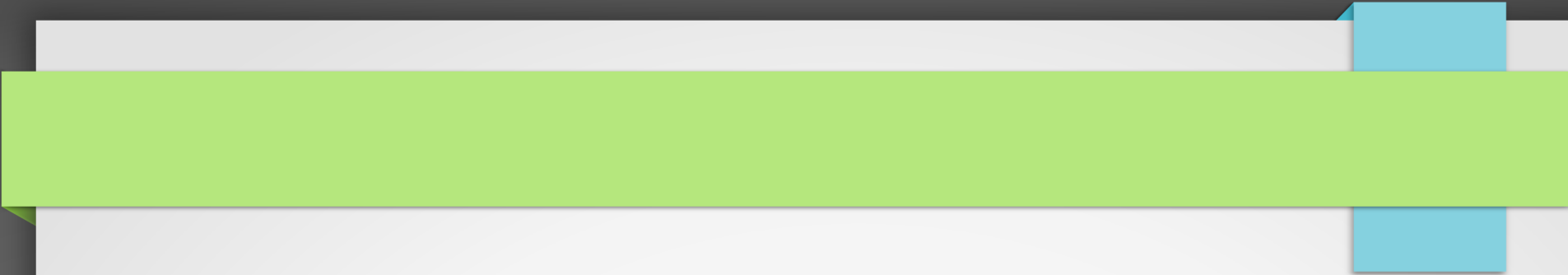
- Refers to extra effort we have to do in future development because of suboptimal design choices.
- Agile development methods aim to manage technical debt through refactoring practices.
- Refactoring is restructuring an existing body of code, altering its internal structure without changing its external behavior.
- Lack of Architecture Agility starts compromising Enhancement Agility.

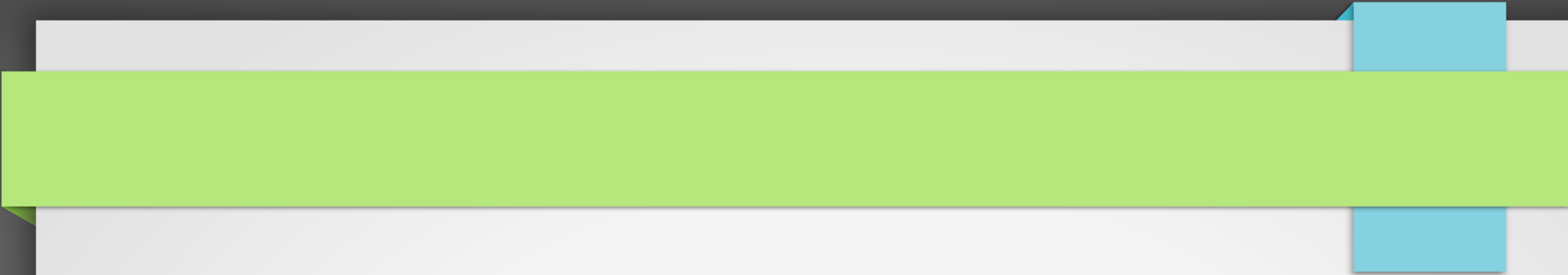
# Informed Anticipation Guiding Agile Release Planning

- Unifying the concepts of technical debt, real options, and uncertainty management is a common focus on the question “Should I take a certain action today in anticipation of increased benefit and reduced cost in the future?”

- 
- Taking the correct action today provides an option which can be acted upon in the future.
  - Identifying architectural elements that enable future stakeholder goals requires mapping options to releases across the lifespan of the system.



- 
- Agile projects focus on stories that are needed in the current release and rely on code-level refactoring to incorporate future stories.
  - However, relying only on code-level refactoring often does not suffice, especially in large-scale development.

- 
- This is where the agile mindset and architecture reasoning tend to diverge.
  - Spending some time architecting can provide better options in many large-scale development contexts that struggle with applying agile techniques.



THANK YOU