INTERNATIONALE HOCHSCHULE

Colloquium

SERGEI ULVIS

STATE MANAGEMEMENT IN FLUTTER APPS

COMPARATIVE STUDY

08 April 2025

CONTENT LIST

01

INTRODUCTION

01 MIN

INTRODUCTION

Flutter: cross-platform framework by Google

Single Dart codebase compiled directly to machine code,
whether Intel x64 or ARM instructions, or to JavaScript if targeting the web

State management: critical for scalability and performance

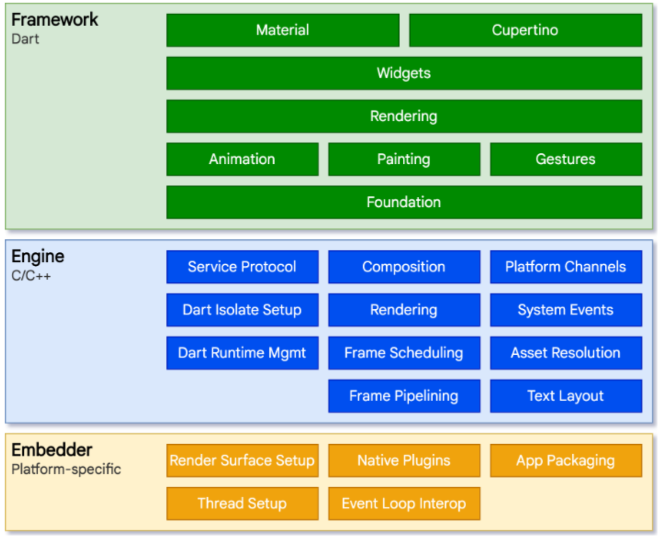
Developers have the freedom to select an approach that best fits
their application's needs

Problem: no mandated solution, fragmented ecosystem

Different state management solutions vary in their structure,
learning curve, and trade-offs

Goal: provide evidence-based guidance for developers

Highlighting the trade-offs of different approaches



02

INTERNATIONALE HOCHSCHULE

OBJECTIVES AND RESEARCH QUESTIONS

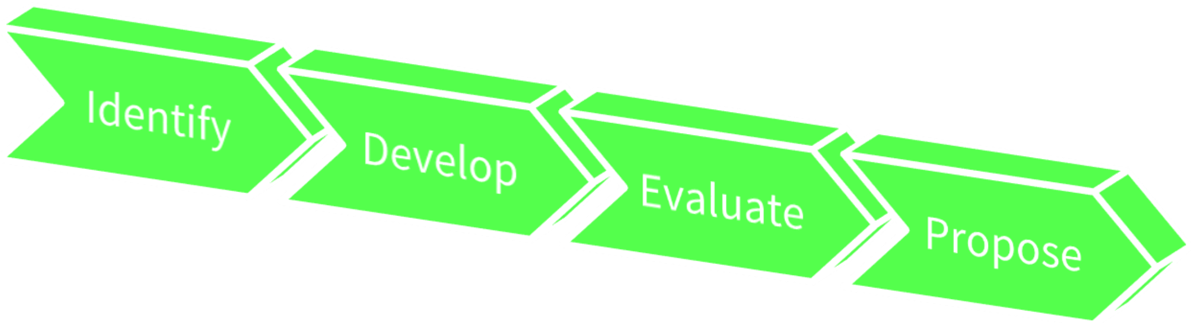
01 MIN

OBJECTIVES AND RESEARCH QUESTIONS

INTERNATIONALE HOCHSCHULE

Objectives:

- Identify popular state management solutions
- Develop prototype applications with consistent functionality using different state management solutions
- Evaluate and compare applications based on key metrics derived from the ISO 25010 standard
- Propose best practices and recommendations to synthesize the findings from the evaluation and provide actionable recommendations for developers



Research questions:

What are the key solutions and their paradigms?

How do solutions perform on key metrics?

What are the best implementation practices?

03

INTERNATIONALE HOCHSCHULE

METHODOLOGY

04 MIN

METHODOLOGY

INTERNATIONALE HOCHSCHULE

- Mixed-method approach, combining literature review with empirical analysis through prototype development and testing
- Literature review: services via IU Library interface (Google Scholar, IEEE, ACM); "Flutter" AND "state management" (broad scope); backward search; inclusion (Eng, modern, >50k downloads on pub.dev) and exclusion criteria
- Literature review identifies key usage methods, strengths, and limitations of both built-in methods and third-party libraries (Provider, Riverpod, BLoC, GetX, Redux, and MobX)
- Evaluation criteria definition based on appropriate (in state management context) sub-characteristics of ISO/IEC 25010:
  - Quantitative measures: time behavior (rebuild counts in widget tests), modularity (Maintainability Index, DCM tool)
  - Qualitative assessments: criteria such as learnability, modifiability, testability and documentation, judged through analysis of code structure and documentation
- Prototype development: E-commerce app (login, product list, cart), same UI and HW environment, different state management
- 6 solutions implemented (Provider, BLoC, Riverpod, GetX, Redux, MobX) and evaluated by defined criteria
- Findings are recorded and analyzed to provide empirical evidence supporting the comparison of state management approaches and conclusions

EVALUATION CRITERIA DEFINITION

iu INTERNATIONALE HOCHSCHULE

- ISO standard 25010: 9 categories (characteristics), 40 sub-characteristics

SOFTWARE PRODUCT QUALITY								
FUNCTIONAL SUITABILITY	PERFORMANCE EFFICIENCY	COMPATIBILITY	INTERACTION CAPABILITY	RELIABILITY	SECURITY	MAINTAINABILITY	FLEXIBILITY	SAFETY
FUNCTIONAL COMPLETENESS	TIME BEHAVIOUR	CO-EXISTENCE	APPROPRIATENESS RECOGNIZABILITY	FAULTLESSNESS	CONFIDENTIALITY	MODULARITY	ADAPTABILITY	OPERATIONAL CONSTRAINT
FUNCTIONAL CORRECTNESS	RESOURCE UTILIZATION	INTEROPERABILITY	LEARNABILITY	AVAILABILITY	INTEGRITY	REUSABILITY	SCALABILITY	RISK IDENTIFICATION
FUNCTIONAL APPROPRIATENESS	CAPACITY		OPERABILITY	FAULT TOLERANCE	NON-REPUDIATION	ANALYSABILITY	INSTALLABILITY	FAIL SAFE
			USER ERROR PROTECTION	RECOVERABILITY	ACCOUNTABILITY	MODIFIABILITY	REPLACEABILITY	HAZARD WARNING
			USER ENGAGEMENT		AUTHENTICITY	TESTABILITY		SAFE INTEGRATION
			INCLUSIVITY		RESISTANCE			
			USER ASSISTANCE					
iso25000.com			SELF-DESCRIPTIVENESS					

Performance efficiency	Interaction capability	Maintainability	Flexibility	Documentation
Time behavior	Learnability	Modularity	Scalability	Documentation
		Analyzability		
		Modifiability		
		Testability		

- 5 criteria (with merged) relevant to state management in Flutter
- + Documentation

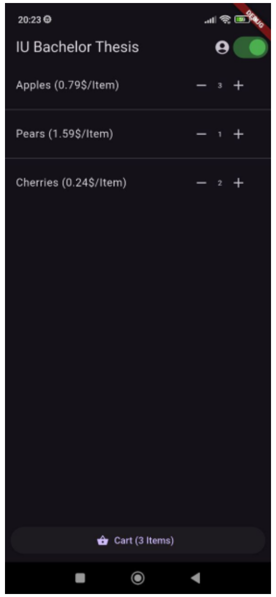
PROTOTYPE DEVELOPMENT

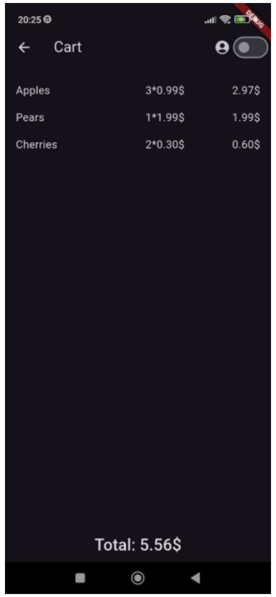
- Implicit requirements:
  - enable a thorough evaluation based on the criteria defined

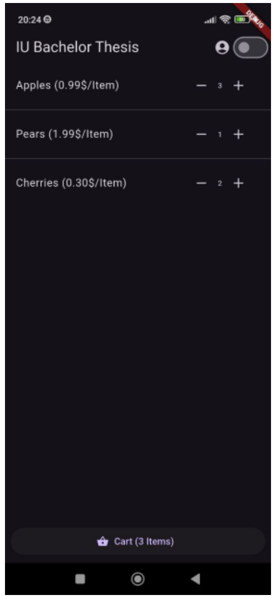
Quantitative measures:
- time behavior: tracking performance via rebuild counts control as reaction on state change
- modularity: ability to calculate Maintainability Index using DCM tool

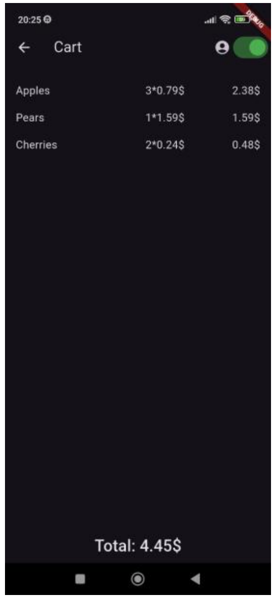
Qualitative assessments: criteria such as
- modifiability: states coupling (entity dependency or interaction, e,g, cost dependency on login state)
- testability: ease of automated testing for state processing implementation (e.g. change items amount)
- learnability: widgets nesting same time with use of separate states dependency (e.g. product_list_screen containing nested UserSwitch and CartButton widgets, which are dependent on different states)

- Explicit requirements:
  - Implement base e-commerce features (user login, products and cart interactions)
  - Clear separation of UI widgets and state classes
  - Navigation between screens with state remaining consistent while navigating
  - Data load from external source









04

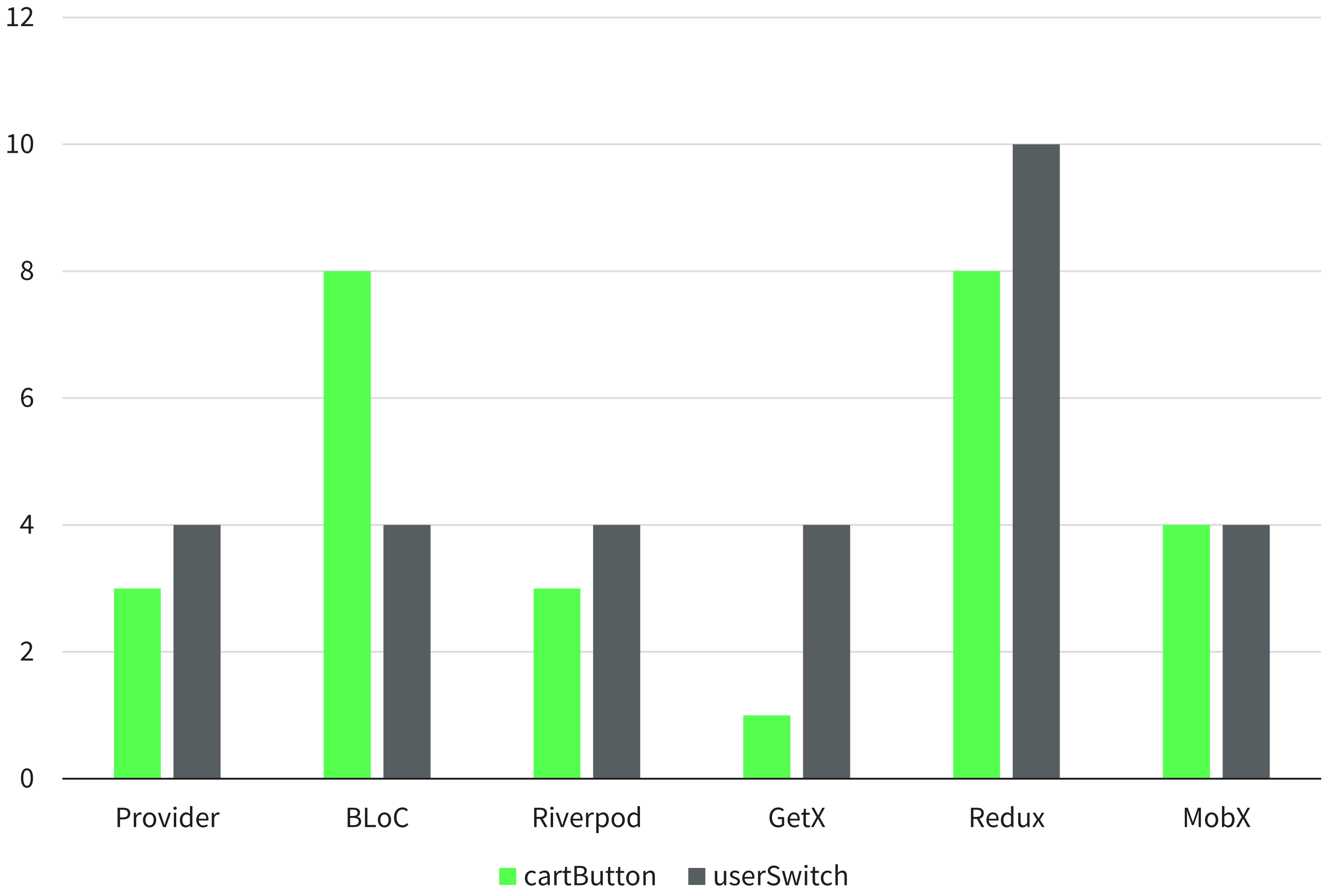
INTERNATIONALE HOCHSCHULE

KEY FINDINGS & CONCLUSION

02 MIN

KEY FINDINGS QUANTITATIVE ANALYSIS

INTERNATIONALE HOCHSCHULE



Time Behavior (widget rebuilds):

- 1st place: GetX: 1/4 - most efficient
- 2nd place: Riverpod/Provider/MobX: 3(4)/4
- 3rd place: BLoC: 8/4
- 4th place: Redux: 8/10 - least efficient

Modularity (Maintainability Index):

- Range: 85.71 (GetX) - 88.35 (BLoC)
- Minimal variation, less discriminative

KEY FINDINGS
QUALITATIVE ANALYSIS

INTERNATIONALE HOCHSCHULE

	Provider	BLoC	Riverpod	GetX	Redux	MobX
Learnability Analyzability	medium	low	high	high	medium	high
Modifiability Scalability	medium	high	high	high	low	high
Testability	high	high	high	high	medium	high
Documentation	medium	high	high	high	high	high

Top performers: Riverpod, GetX, MobX
Provider: medium scalability, simple but limited

BLoC: high scalability, low learnability
Redux: low scalability, poor fit for Flutter

CONCLUSION

INTERNATIONALE HOCHSCHULE

01

Criteria Impact

- Most metrics (e.g., time behavior) revealed strengths & weaknesses
- *Exception*: Maintainability Index (85.71–88.35) showed little variation

02

Notable Findings

- **Provider**: Simple but limited scalability (medium rating)
- **BLoC**: Scalable, testable, but low learnability (cartButton: 8)
- **Redux**: Lowest scalability, poor efficiency (cartButton: 8, userSwitch: 10)

03

Top Performers

- **Riverpod**: High learnability, scalability, flexibility in provider design
- **GetX**: High efficiency (cartButton: 1), minimal boilerplate, reactive optimization
- **MobX**: High clarity, balanced efficiency (cartButton: 4, userSwitch: 4)

04

Key Takeaways

- **Riverpod**: Evolves Provider with robustness
- **GetX**: Excels in performance & simplicity
- **MobX**: Strong in structured reactivity

05

INTERNATIONALE HOCHSCHULE

RECOMMENDATIONS

01 MIN

RECOMMENDATIONS

INTERNATIONALE HOCHSCHULE

GetX

- **Best For**: Apps of all sizes prioritizing performance & rapid development
- **Use Case**: Real-time messaging, e-commerce (dynamic UI updates)
- **Strengths**:
  - Time Behavior: cartButton: 1, userSwitch: 4
  - High learnability, modifiability, scalability, testability, documentation
  - Reactive (Obx), minimal rebuilds, centralized controllers
- **Consideration**: Needs discipline to avoid state sprawl in massive projects

Provider

- **Best For**: Small apps valuing simplicity & quick setup
- **Use Case**: To-do lists, prototypes
- **Strengths**:
  - Time Behavior: cartButton: 3, userSwitch: 4
  - High testability, medium learnability, medium documentation
  - Familiar Flutter concepts, lightweight
- **Consideration**: Limited scalability (ProxyProvider cap, nesting issues)

Riverpod

- **Best For**: Medium-sized apps needing flexibility & scalability
- **Use Case**: Social media platforms (profiles, feeds)
- **Strengths**:
  - Time Behavior: cartButton: 3, userSwitch: 4
  - High learnability, modifiability, scalability, testability, documentation
  - Intuitive ref-based access, dynamic providers
- **Consideration**: Less reactive than GetX, but more structural freedom

Secondary Options:

- **BLoC**: High scalability, complex projects, low learnability (cartButton: 8)
- **Redux**: Low scalability, poor efficiency (userSwitch: 10)
- **MobX**: High ratings, balanced efficiency (cartButton: 4), large apps

Why MobX is in Secondary Options: due to code auto generation –
In complex situations that may lead to unclear behavior,
which may become the blocker for less experienced developers

06

OUTLOOK

01 MIN

OUTLOOK

INTERNATIONALE HOCHSCHULE

01

REPLACE MI

Maintainability Index's narrow range (85.71 - 88.35) suggests exploring alternative metrics

02

INCLUDE EMERGING LIBRARIES

Like June, Binder, or revisit excluded ones

03

REDUX REVIEW

Review Redux with developers working across frameworks

04

CHECK GETX IN HIGH LOAD

Check GetX productivity with larger datasets or real-world application

05

DEVELOPERS SURVEYS

Assessing developer experience through surveys on onboarding ease or satisfaction would complement technical findings

INTERNATIONALE HOCHSCHULE

THANK YOU

Q&A SESSION

Sergei Ulvis
+7 ....
sergei.ulvis@gmail.com