On the Role of Fitness Dimensions in API Design Assessment

An Empirical Investigation

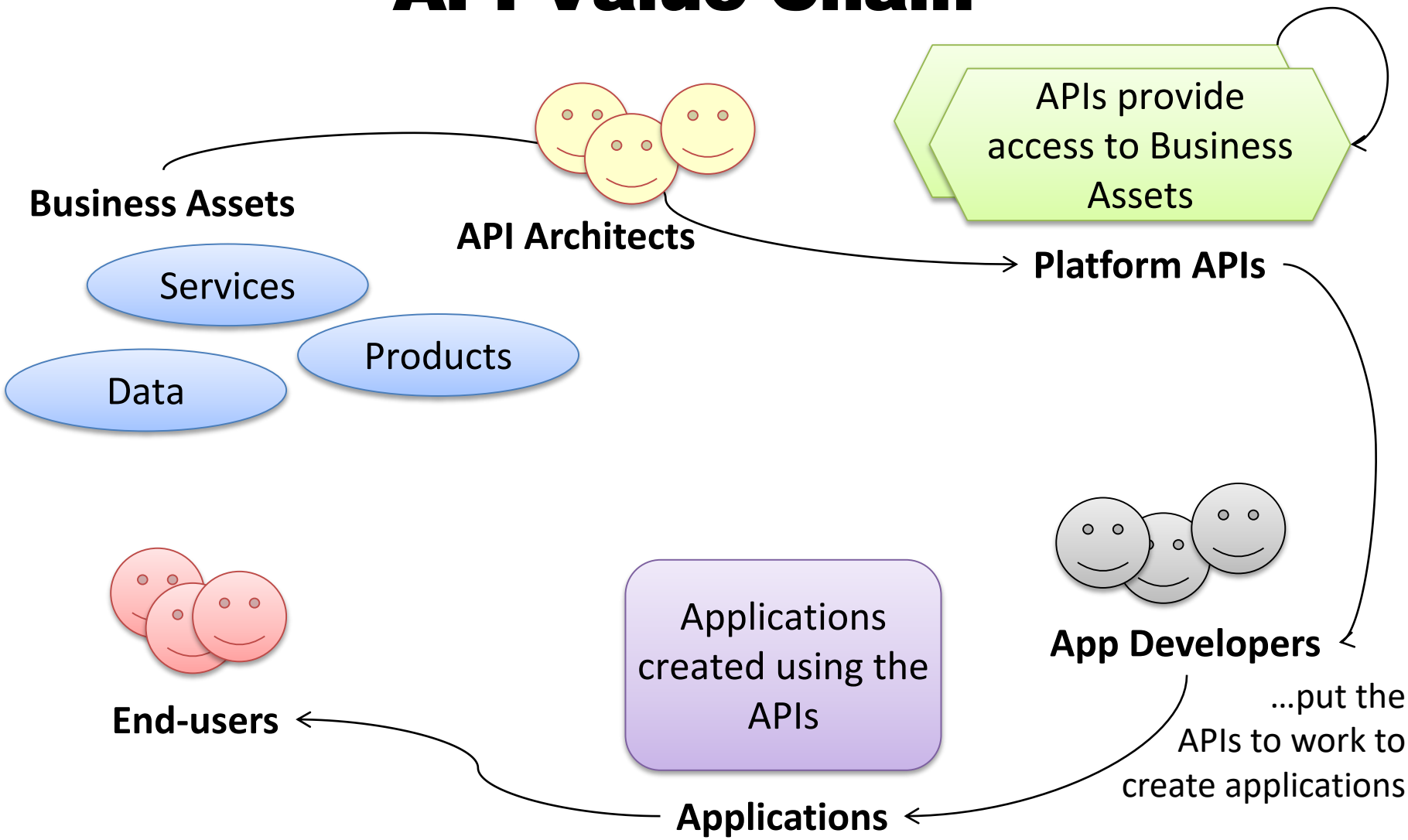
Amir Zghidi[1], Imed Hammouda[2], Brahim Hnich[3] and Eric Knauss[2]

[1] *Independent Researcher*

[2] *Chalmers | University of Gothenburg, Sweden*

[3] *University of Monastir, Tunisia*

API Value Chain



Research Objective

What factors to consider when applying fitness dimensions in assessing API design?

Research Approach

- A Swedish software intensive company operating in the embedded systems domain.
- Qualitative and quantitative data regarding three different software platforms embedded in a hardware device.
- Qualitative data:
  - Workshop & focus group discussions
  - API architects, product managers, internal API users, and technical leaders
- Quantitative data
  - Survey
  - Eight API architects and one external API user

23 Fitness Dimensions

Working Framework

Consistency

Viscosity

Compatibility

Synchrony

Progressive Evaluation

Testability

Abstraction Level

Atomic Setting

Role Expressiveness

Premature Commitment

Penetrability

Easiness to Use

API Latency

Learning Style

API Evolvability

Error Checking and Responsiveness

Model Complexity

Functional Completeness

Work Step Unit

Domain Correspondence

API Elaboration

Security

Abstraction Level

This fitness dimension describes the levels of abstraction exposed by the API.

Low

High

Many API components are needed for implementing a particular developer goal

Only one API component is needed for implementing a particular developer goal

Learning Style

This fitness dimension describes the degree of the learning requirements posed by the API.

Low — High

No learning requirements are needed for implementing developer goals

Deep learning requirements are needed for implementing developer goals

Working Framework

This fitness dimension describes the size of the conceptual chunk needed to work effectively with the API.

Low

Only API level information is needed for implementing developer goals

High

System assets/resources are needed for implementing developer goals

Premature Commitment

This fitness dimension describes the extent to which a developer has to make decisions before all the needed information is available to achieve a goal.

Low	High
Late binding developer decisions	Early binding developer decisions

Consistency

This fitness dimension describes how much of the rest of the API can be inferred once part of the API is learned for implementing a given goal.

Low

API does not use the same design patterns and idioms at all

High

API uses the same design patterns and idioms

Penetrability (also known as capabilities)

This fitness dimension describes how the API facilitates exploration, analysis and understanding of its components and system capability/properties when implementing developer goals.

Low — High

API reveals minimal details about system capability/properties

API gives enough information to allow the developer to understand the intricate working details of the API and system capability/properties

API Elaboration

This fitness dimension describes the extent to which the API must be adapted to meet the needs of a targeted developer goal.

Low

The types exposed by the API can be used without requiring further elaboration

High

API requires replacing or introducing whole new types

Role Expressiveness

This fitness dimension describes how apparent the relationship is between each component/part and the program as a whole when developing a goal.

Low

Code cannot be interpreted correctly and does not fully match developer's expectations

High

Code can be interpreted correctly and highly matches the developer's expectations

Domain Correspondence

This fitness dimension describes how clearly the API components map to the domain when implementing developer goals.

Low

API types do not map directly on to the domain concepts even after describing the mapping

High

Types exposed by the API map directly on to the domain concepts

Work Step Unit

This fitness dimension describes the amount of work the developer has to do to accomplish a goal, in terms of the progression from starting the work to completing the goal.

Low	High
Code contained within one local code block	Code contained within multiple code blocks

Progressive Evaluation

This fitness dimension describes the extent to which partially completed code can be executed to obtain feedback on code behavior when implementing developer goals.

Low — High

Developer needs to work with multiple API components/blocks in parallel to get feedback

Feedback is provided after each line of code written

API Latency

This fitness dimension describes the amount of real-time delay lies in between requests and responses when using the API to accomplish developer goals.

Low	High
Immediate response	Lots of work in the background

Synchrony

This fitness dimension describes whether the API responds synchronously or asynchronously to requests corresponding to developer goals.

Low — High

API respond asynchronously

API responds synchronously

Compatibility

This fitness dimension describes how robust is the API for matching different versions of developer goals to different version of the platform.

Low	High
API versions are extremely incompatible	API versions are highly compatible

Model Complexity

This fitness dimension describes the kind of API model that the API implies as perceived when implementing developer goals.

Low	High
The model behind the API is simple	The model behind the API is complex

API Evolvability

This fitness dimension describes how well the API is prepared for increased/diverged functionality in the future to support new developer goals.

Low — High

API is rigid with regard to incorporating future changes

API can easily acquire novel functionality

Testability

This fitness dimension describes the degree to which the API supports testing in a given test context of developer goal implementation.

Low	High
Limited support for testing applications	Systematic and comprehensive support for testing applications

Error Checking and Responsiveness

This fitness dimension describes the ability of the API to report on the status and progress of the developer goal as it is being developed, in particular relating to checking errors and responding accordingly

Low	High
API is totally silent on the state of application development	API is highly chatty when it comes to messages regarding application development

Viscosity

This fitness dimension describes the ease at which a developer can make changes to existing application code for a goal.

Low — High

API allows developers to make changes to code written against an API easily

API allows developers to make changes to code written against an API with significant effort

Atomic Setting

This fitness dimension describes the ability of the API to handle requests as atomic operations when implementing developer goals.

Low

API does not provide support for encapsulating requests as atomic operations

High

API provides comprehensive support for processing, executing and canceling requests as atomic operations

Security

This fitness dimension describes the extent to which the API provides support to protect developer goals from unauthorized access, use, disclosure, disruption, modification, or destruction.

Low

API has no capability for controlling untrusted code

High

API has strong built-in mechanisms for detecting, resisting, reacting to, and recovering from security attacks

Easiness to Use

This fitness dimension describes to the extent to which the API can be used to achieve developer goals without referring to other supporting resources (i.e. API documentation).

Low

It is difficult to use the API without the support of other resources

High

API can be effectively used without the need for other supporting resources

Functional Completeness

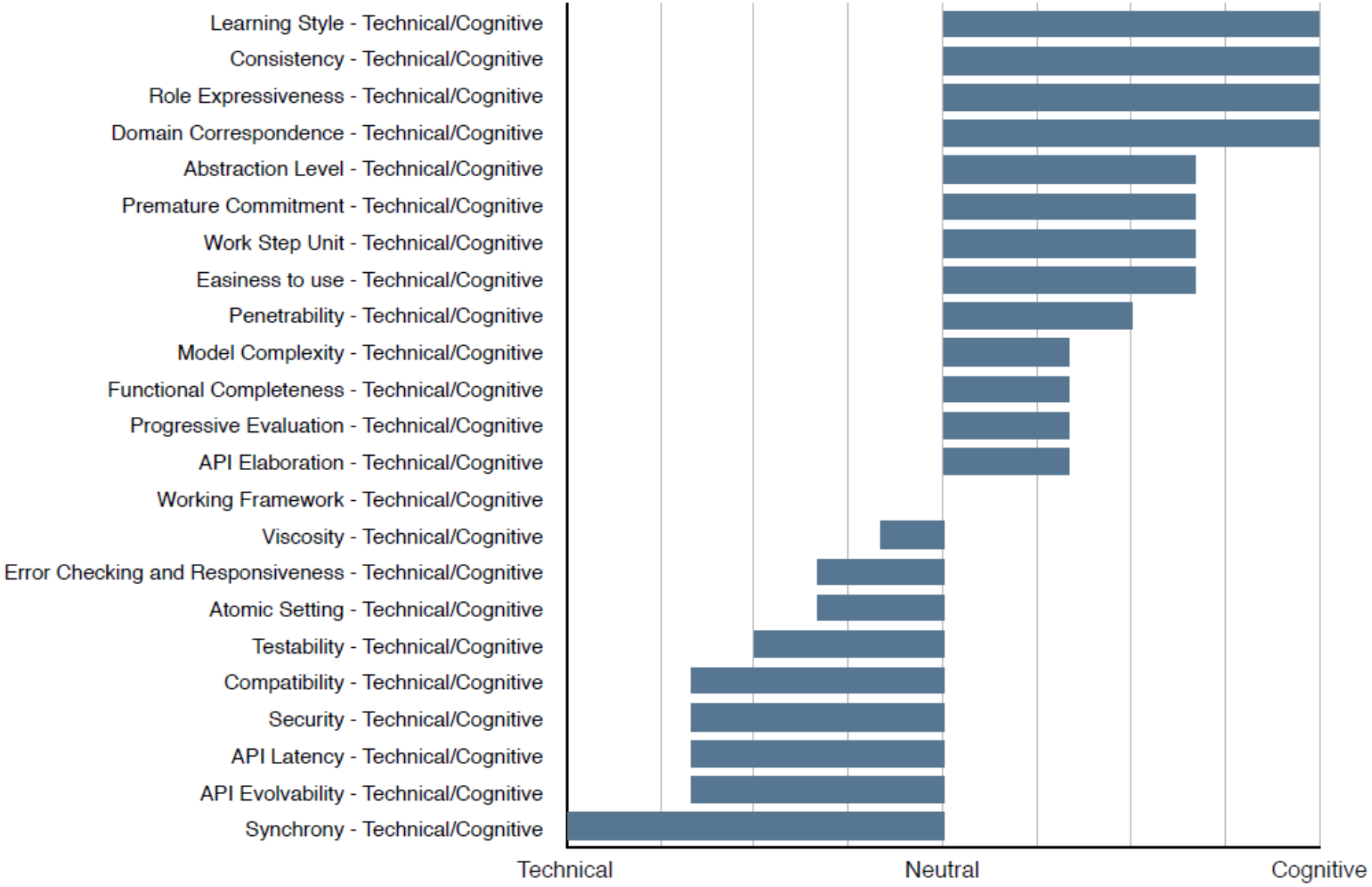
This fitness dimension describes the degree to which the API supports the implementation of all developer goals

Low — High

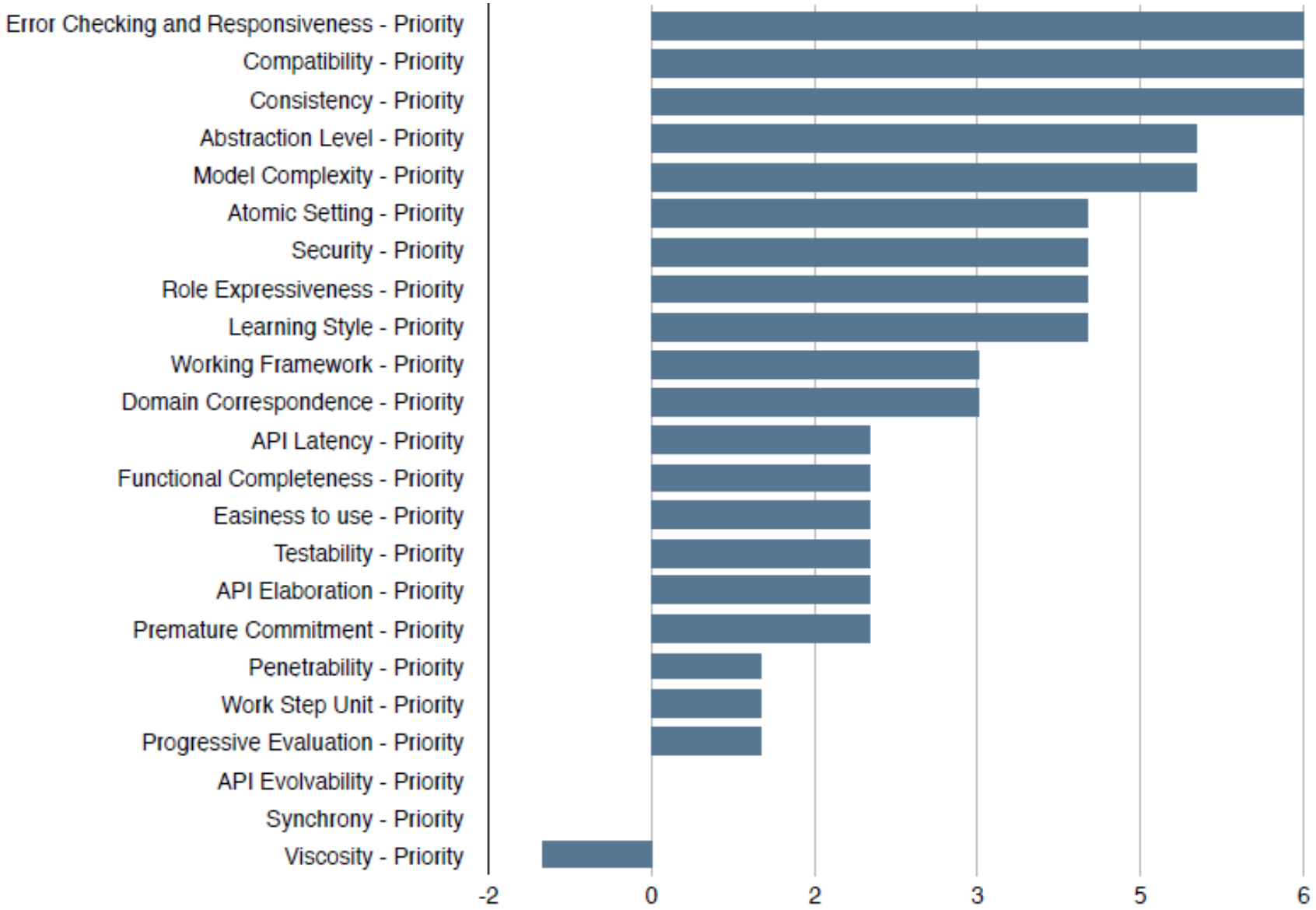
API can be used to achieve a small part of user goals

API can be used to achieve all developer goals

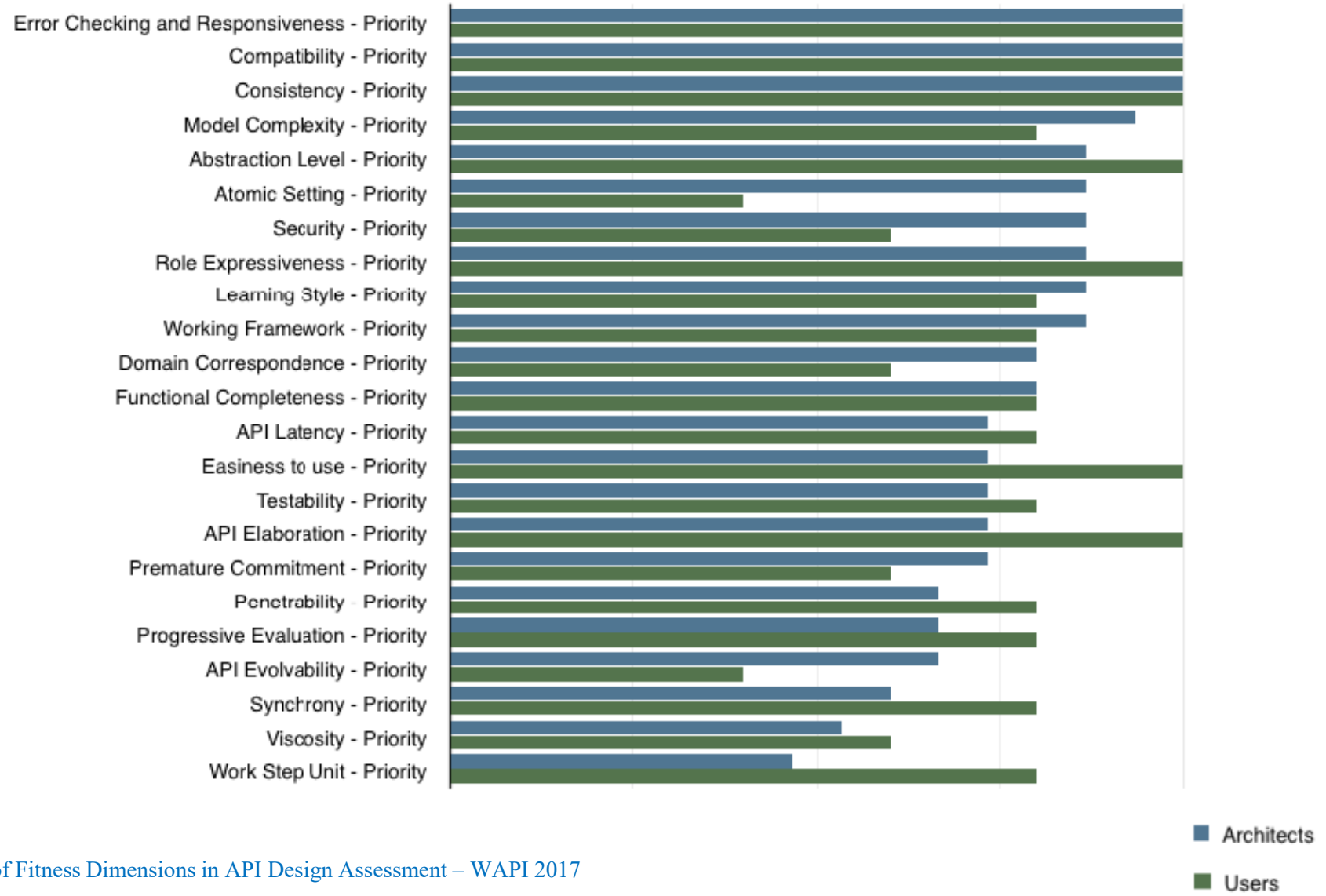
Fitness Dimensions - Classification



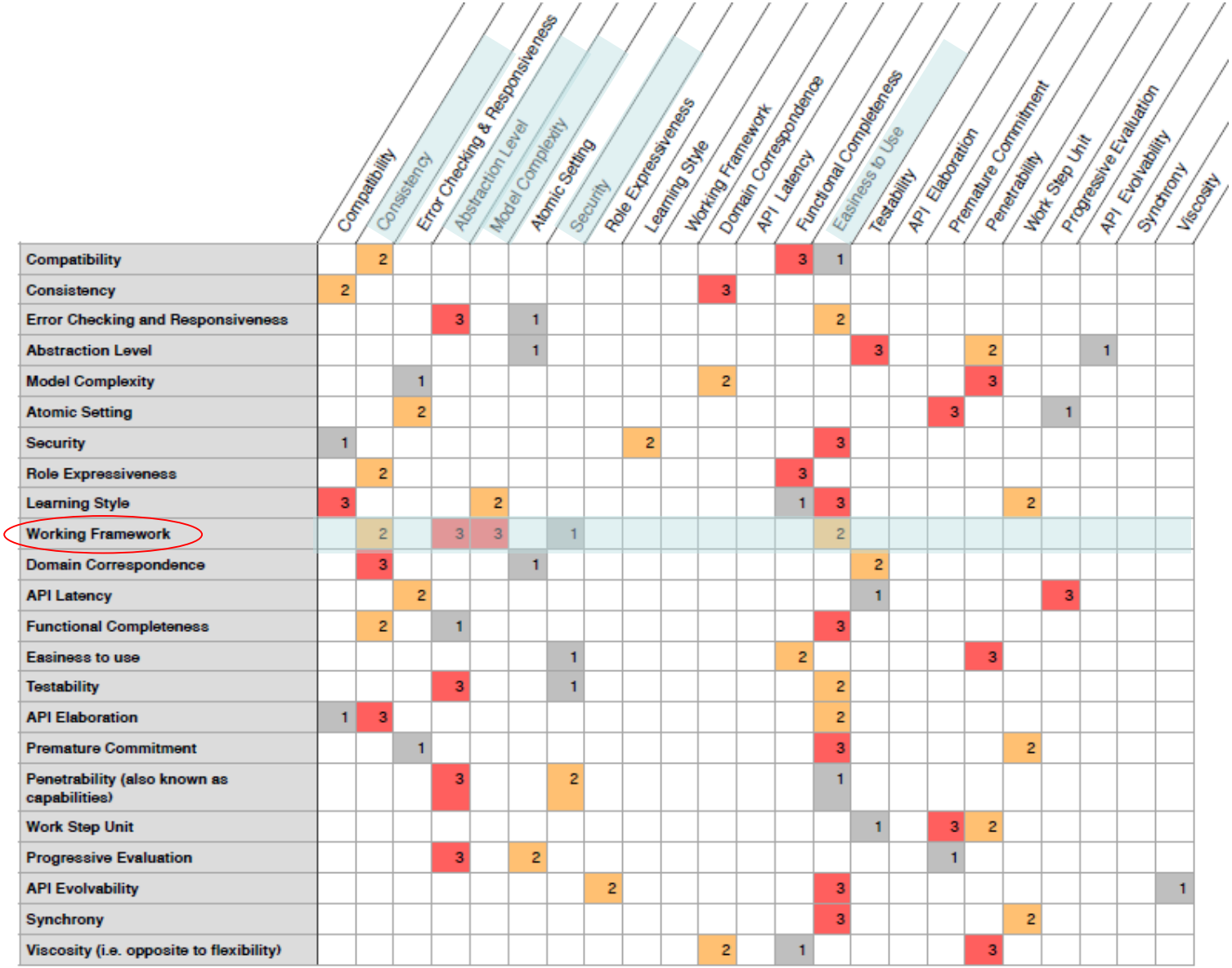
Fitness Dimensions – Priority (API Architects)



Fitness Dimensions – Priority (Architects vs Users)



Fitness Dimensions - Tradeoffs



	Compatibility	Consistency	Error Checking & Responsiveness	Abstraction Level	Model Complexity	Atomic Setting	Security	Role Expressiveness	Learning Style	Working Framework	Domain Correspondence	API Latency	Functional Completeness	Easiness to Use	Testability	API Elaboration	Premature Commitment	Penetrability	Work Step Unit	Progressive Evaluation	API Evolvability	Synchrony	Viscosity
Compatibility		2											3	1									
Consistency	2										3												
Error Checking and Responsiveness				3		1								2									
Abstraction Level						1									3		2			1			
Model Complexity			1								2						3						
Atomic Setting			2													3			1				
Security	1								2					3									
Role Expressiveness		2											3										
Learning Style	3				2								1	3				2					
Working Framework		2		3	3		1							2									
Domain Correspondence		3				1									2								
API Latency			2												1					3			
Functional Completeness		2		1										3									
Easiness to use							1						2				3						
Testability				3			1							2									
API Elaboration	1	3												2									
Premature Commitment			1											3				2					
Penetrability (also known as capabilities)				3			2							1									
Work Step Unit															1		3	2					
Progressive Evaluation				3		2											1						
API Evolvability								2						3									1
Synchrony														3				2					
Viscosity (i.e. opposite to flexibility)											2		1				3						

Working with Personas

Persona 1

Client Application Developer

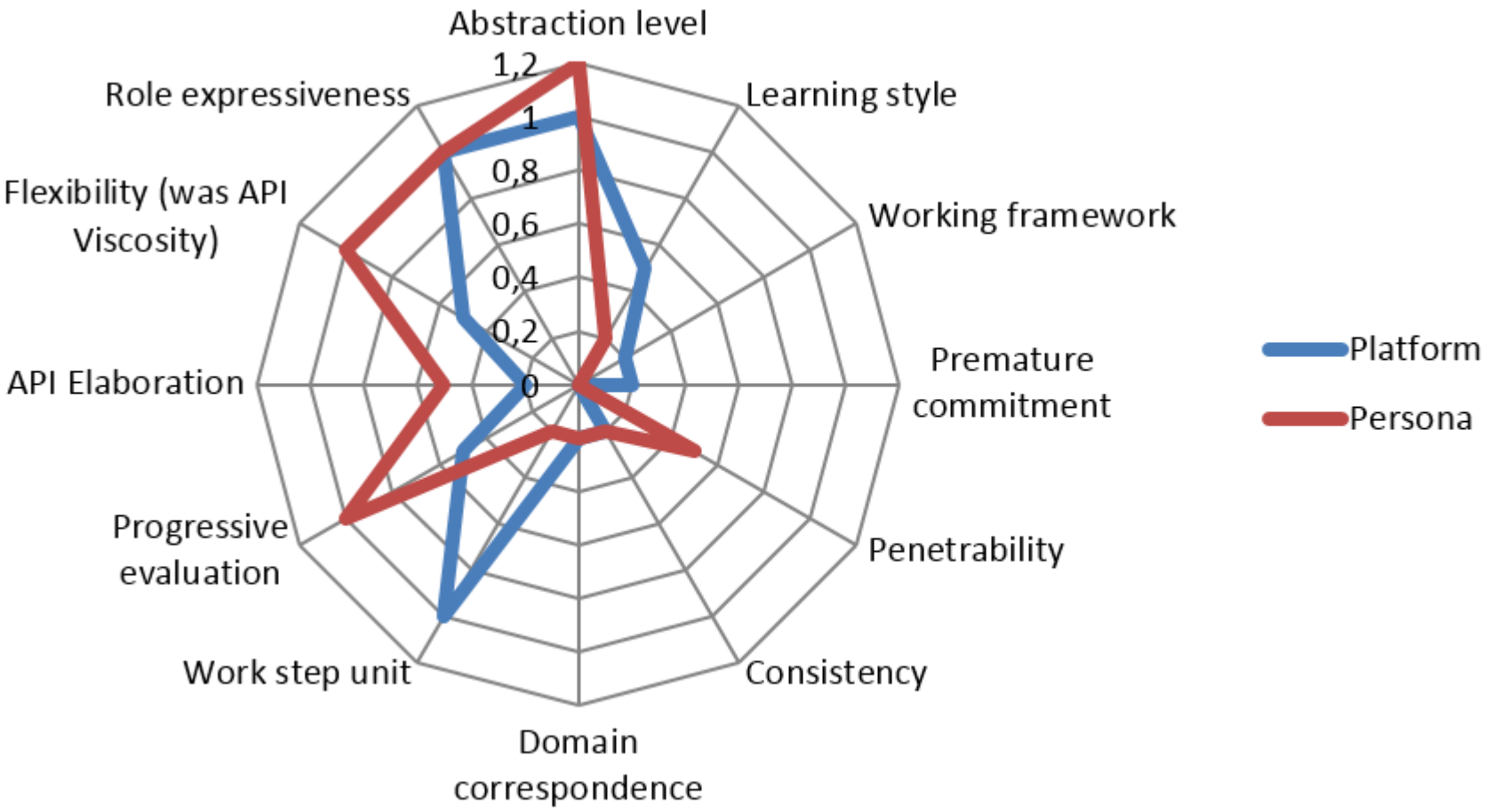
- Opportunistic developer
- E.g. needs new functionality via the platform
- Develops both client and the platform
- E.g. Company X
- Priority: 3

Persona 2

Feature External Developer

- Systematic developer
- Adapts to existing systems for business reasons
- E.g. Company Y
- Priority: 4

Platform API Fitness



Discussion Points

- How to formalize API assessment?
- How to balance between the different preferences of stakeholders involved?


Thank you!
Questions?