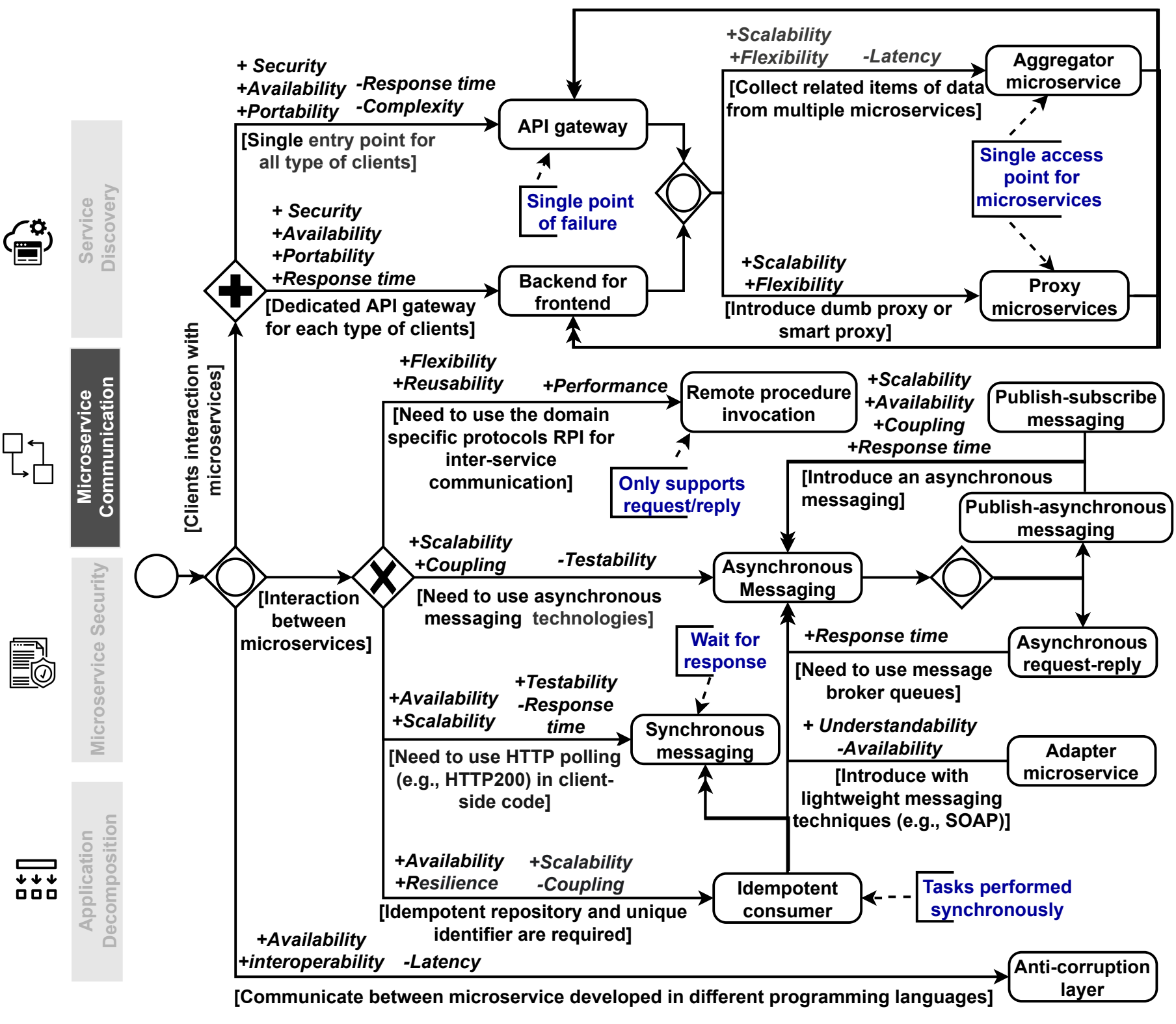
Service Discovery
Microservice Communication
Microservice Security
Application Decomposition
[Clients interaction with microservices]
+ Security
+Availability
+Portability
-Response time
-Complexity
API gateway
[Single entry point for all type of clients]
Single point of failure
+ Security
+Availability
+Portability
+Response time
Backend for frontend
[Dedicated API gateway for each type of clients]
+Scalability
+Flexibility
-Latency
Aggregator microservice
[Collect related items of data from multiple microservices]
Single access point for microservices
+Scalability
+Flexibility
Proxy microservices
[Introduce dumb proxy or smart proxy]
+Flexibility
+Reusability
+Performance
Remote procedure invocation
[Need to use the domain specific protocols RPI for inter-service communication]
Only supports request/reply
+Scalability
+Availability
+Coupling
+Response time
Publish-subscribe messaging
[Introduce an asynchronous messaging]
Publish-asynchronous messaging
[Interaction between microservices]
+Scalability
+Coupling
-Testability
Asynchronous Messaging
[Need to use asynchronous messaging technologies]
Wait for response
+Response time
Asynchronous request-reply
[Need to use message broker queues]
+Availability
+Scalability
+Testability
-Response time
Synchronous messaging
[Need to use HTTP polling (e.g., HTTP200) in client-side code]
+ Understandability
-Availability
Adapter microservice
[Introduce with lightweight messaging techniques (e.g., SOAP)]
+Availability
+Resilience
+Scalability
-Coupling
Idempotent consumer
Tasks performed synchronously
[Idempotent repository and unique identifier are required]
+Availability
+interoperability
-Latency
Anti-corruption layer
[Communicate between microservice developed in different programming languages]