Category 1: Design

Smell 1: Use of business logic in communication among services

Definition： If the communication layer contains business logic means it has this smell such as data transfer in the communication layer.

Design Violation： Single Responsibility Principle

Interest: There will be additional maintenance costs and changes in the service logic will cause the communication layer to change as well. And communication team must understand the details in the logic of services.

Solution: Remove the business logic from the communication layer and each service takes responsibility for business logic.

Similar Smells: /

Involved papers: P1, P7

Smell 2： Mega Microservice

Definition： One microservices takes responsibility for any concerns

Design Violation： Single Responsibility Principle

Interest: It will cause difficult in maintaining, test and complexity of software.

Solution: Decompose the mega service into smaller microservices that just take responsibility for only one concern.

Similar Smells: Coarse Service

Involved papers: P2, P4, P5, P6, P9, G8

Smell 3： Nano Microservice

Definition: The granularity of a monolith system divided into microservices is too fine that a single microservice does not fulfill one business capability.

Design Violation： Single Responsibility Principle

Interest: The principle for monolith to microservice is based on business capability and suitable granularity. It will cause the coupling of the services.

Solution： Decompose a monolith around business capabilities and microservices should take responsibility for one concern.

Similar Smells: Excessive number of small products, Grinding Dusty Services

Involved papers: P1, P4, P9, P13, G3

Smell 4: Cyclic Dependencies

Definition: The dependencies or calls of microservices are like a cycle.

Design Violation： Indendpence, Acyclic dependencies principle

Interest: The microservices are not independent and if one fail will cause other microservices to fail too.

Solution: Avoid cyclic dependencies by grouping strongly dependent microservices into a single microservice and using API to avoid the direct calls of microservices.

Similar Smells:/

Involved papers: P3, P4, P5, P9, P11, P12, G1

Smell 5: Wrong Cuts
Definition: The principle for dividing microservices should base on business capability and microservice just focus on one single concern. If dividing microservice does not follow the principle will cause the smell.
Design Violation: Single Responsibility Principle
Interest: It will cause the complexity and high coupling of microservice. It is harder to maintain the microservices.
Solution: Decompose the microservice-based on the business capability.
Similar smells: /
Involved papers: P4, P5, P9, P11, P12, G2, G6, G7, G9

Category 2: Deployment
Smell 6: Wobbly service interactions
Definition: Microservice needs to be independent and clear boundaries of others. If the interaction of microservices is wobbly that means it is this smell.
Design Violation: Independence
Interest: If one microservice fails will cause another one to fail too and cause the coupling of different microservices.
Solution: The most common solution is the usage of a circuit breaker to wrap the invocations from a microservice to another.
Similar Smells: Microservice coupling
Involved papers: P1, P6, P8, G5

Smell 7: Shared Libraries
Definition: Microservices share runtime libraries or execution files.
Design Violation: Decentralisation
Interest: It causes the coupling of different microservices.
Solution: Runtime assets should not be shared and they need to belong to the corresponding microservice.
Similar smells: I was taught to share
Involved papers: P3, P4, P5, P6, P7, P9, P12, P13, G4, G5

Smell 8: Shared Persistence
Definition: Multi microservices share one database.
Design Violation: Decentralisation
Interest: It will cause the coupling of different microservices and harder to maintain them.
Solution: Separate the data into different databases that are only accessed by corresponding microservices.
Similar smells: /
Involved papers: P1, P4, P5, P9, P10

Category 3: Monitor & log
Smell 9: Insufficient message traceability
Definition: When messages contain insufficient data that cause difficulty to find the source of the messages.
Design Violation： High availability
Interest: It will cause harder to find dependencies of microservices.
Solution: Add ownership metadata into messages.
Similar smells: /
Involved papers: P1, P13

Smell 10: Manual Configuration
Definition: Configuration of instances, services and hosts is done manually by developers.
Design Violation： Auto-provisioning
Interest: Manual configuration is time-consume and error-prone
Solution: Configuration automatically by tools.
Similar smells: /
Involved papers: P1, P4, P13, G5

Smell 11: Dismiss Documentation
Definition: Inconsistent documents or outdated documents for software will cause this smell
Design Violation： High availability
Interest: With the development of microservices, if the documents of API are dismissed that will hinder the cooperation of different teams.
Solution: Ensure each team is responsible for maintaining a published API for their service.
Similar Smells: /
Involved papers: P2

Smell 12: No health check
Definition: No endpoints for checking the health of microservice.
Design Violation： High availability
Interest: Consumers may wait a long time to get a response from microservices that is down.
Solution: Add endpoints for checking microservice health.
Similar Smells: /
Involved papers: P4

Smell 13: Local Logging
Definition: The information or logs of microservices are stored in local storage.
Design Violation： Auto-provisioning
Interest: The logs in local are difficult to analyze and monitor.
Solution: Use logging systems to store the microservices logs and information.
Similar Smells: /
Involved papers: P4, P9, P13

Category 4: Communication
Smell 14: Lack of communication standards among microservices
Definition： Lack of API or messages format for microservices. Each team has its standards for communication.
Design Violation： Scalability
Interest: Need cost more for transform the messages and add the complexity of software.
Solution: Define same or consist standards for communication of different teams.
Similar Smells： /
Involved papers: P1, P7

Smell 15: HardCoded Endpoints
Definition: The IP addresses, ports, and endpoints of microservices are explicitly/directly specified in the source code.
Design Violation： Scalability
Interest: It is difficult to track the URLs and endpoints and when the ports change the deploying of microservices are also needs to change.
Solution: Use service discovery: client-side service discovery or server-side service discovery
Similar Smells:/
Involved papers: P3, P4, P5, P6, P8, P9, P11, P13

Smell 16: No API Gateway
Definition: Microservices are exposed and consumers communicate with them directly.
Design Violation： Scalability
Interest: Consumers need to know each microservices in detail and harder to maintain the endpoints of microservices.
Solution: Use API Gateway as the entry for communication.
Similar Smells: /
Involved papers: P4, P5, P6, P7, P8, P9, P12, P13

Smell 17: Timeouts
Definition: The unsuitable time set for sending messages or waiting for a response.
Design Violation： High Availability
Interest: Too short timeout will cause no enough time to handle the request and too long time will waste the time to wait for unavailable microservices.
Solution: Use circuit breakers, which are interceptors that check regularly for service "health" (availability and responsiveness), and decline requests automatically if the microservice is down.
Similar Smells: /
Involved papers: P4, P13

Smell 18: No API Versioning
Definition: No information is available on the microservice version. The microservice should support multi API versions including the new and the old ones.
Design Violation： Scalability

Interest: Changes to a microservice API will impact all consumers or consumers can not communicate microservice using different API versions.
Solution: The same microservice should support multi API versions.
Similar Smells: /
Involved papers: P4, P5, P9, P11, P13, G3, G5

Smell 19: ESB misuse
Definition: Central ESB is used for connecting microservices in an application
Design Violation： Scalability
Interest: ESB abuse may lead to undesired centralization of business logic and dumb services and coupling of microservices
Solution: Use API to replace the ESB or resize the ESB.
Similar Smells: /
Involved papers: P5, P6, P9

Category: Team & Tool
Smell 20: Single Layer Team
Definition: One team takes responsibility for many services. Each team has no explicit responsibility boundary for services.
Design Violation： Single Responsibility Principle and Independence
Interest: It destroys the independence among services and causes external communication cost among teams.
Solution: Divided the teams that take responsibility for different services. Each team just takes responsibility for one service.
Similar Smells: /
Involved papers: P2, P6, P9, G2, G10

Smell 21: Inadequate techniques support
Definition: The techniques or tools for Microservice are not enough and dismiss a lot of key points of Microservice.
Design Violation： High availability
Interest: The automation of the service is broken and most of its advantages are not demonstrated.
Solution: Take the DevOps culture and automation into consideration. And use some specific tools for supporting Microservices.
Similar Smells: Single DevOps Toolchain
Involved papers: P1, P2, P4, P9, P13, G2, G5

Smell 22: Too many Standards
Definition: In one program, many developing languages or frameworks are used. Each team uses their own techniques and no specific standard to limit them.
Design Violation： High availability
Interest: Add unnecessary complexity and maintenance problems
Solution: Predefine the standards for the program and select the most familiar developing language and frameworks.

Similar Smells: /

Involved papers: P1, P5, P7, P9, P12, P13

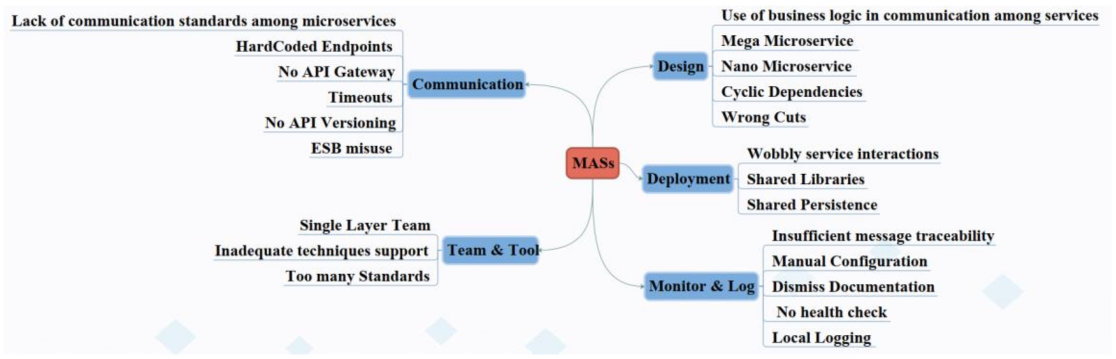

Figure: The categories for 22 MASs