


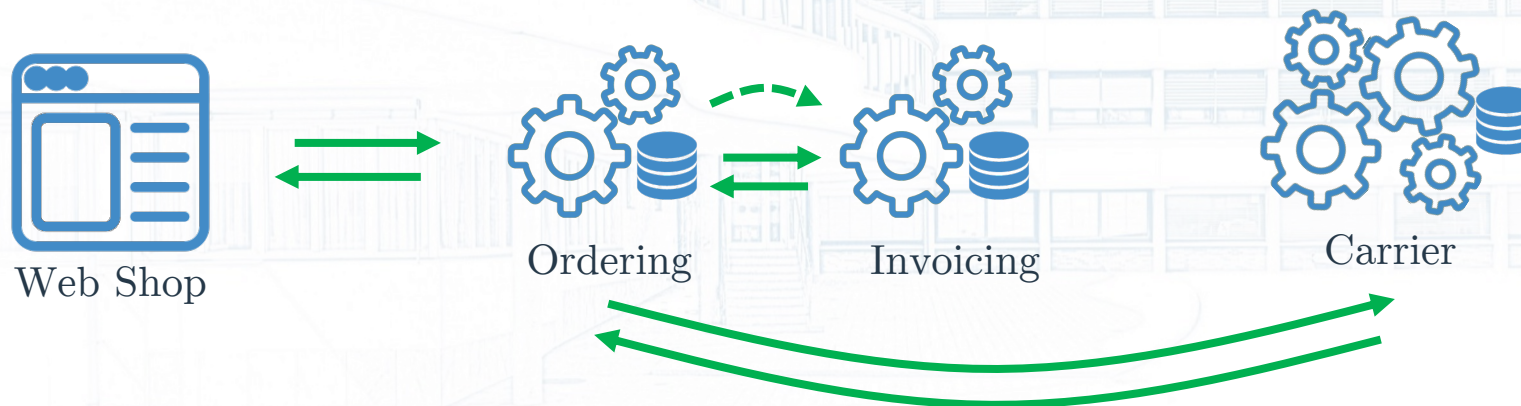
Microservice API Evolution in Practice: A Study on Strategies and Challenges

Alexander Lercher, Johann Glock, Christian Macho, Martin Pinzger
University of Klagenfurt, Austria
{alexander.lercher, johann.glock, christian.macho, martin.pinzger}@aau.at

Microservice APIs

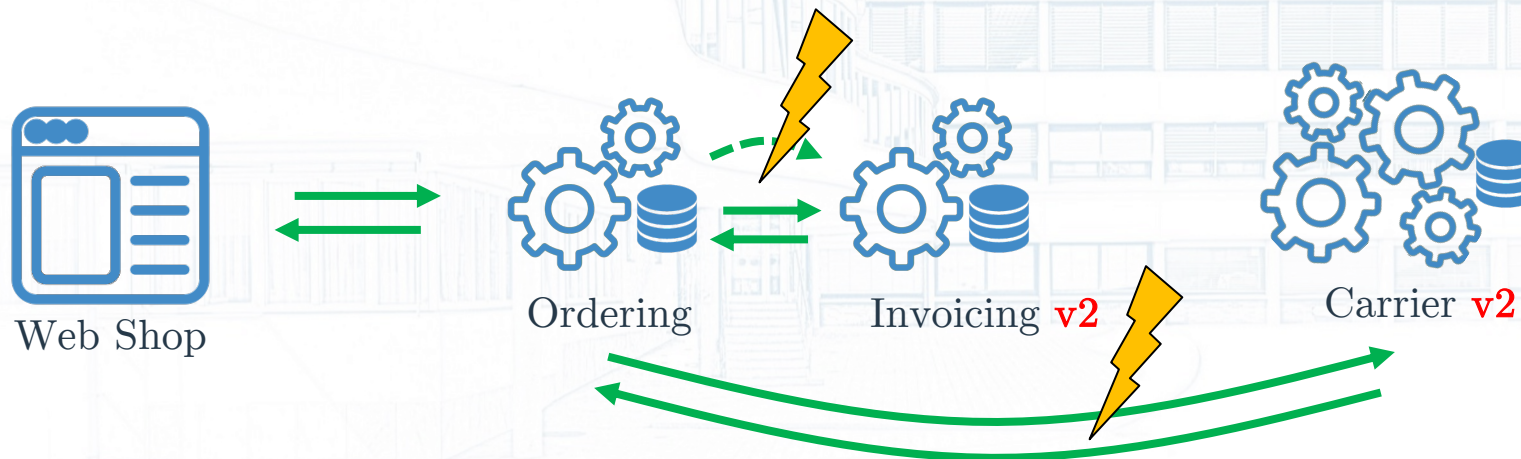
Microservice APIs

- Request-response via HTTP REST 
- Event-driven via message brokers 



Microservice API Evolution

- Request-response via HTTP REST 
- Event-driven via message brokers 

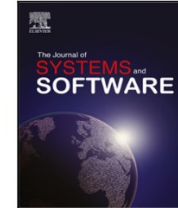




Contents lists available at [ScienceDirect](https://www.sciencedirect.com)

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss



Microservice API Evolution in Practice: A Study on Strategies and Challenges[☆]

Alexander Lercher^{*}, Johann Glock, Christian Macho, Martin Pinzger

Department of Informatics Systems, University of Klagenfurt, Universitätsstraße 65-67, Klagenfurt, 9020, Austria

ARTICLE INFO

Keywords:

Microservice architecture
API evolution
API versioning
Backward compatibility
API design degradation
Development team collaboration

ABSTRACT

Nowadays, many companies design and develop their software systems as a set of loosely coupled microservices that communicate via their Application Programming Interfaces (APIs). While the loose coupling improves maintainability, scalability, and fault tolerance, it poses new challenges to the API evolution process. Related works identified communication and integration as major API evolution challenges but did not provide the underlying reasons and research directions to mitigate them. In this paper, we aim to identify microservice API evolution strategies and challenges in practice and gain a broader perspective of their relationships. We conducted 17 semi-structured interviews with developers, architects, and managers in 11 companies and analyzed the interviews with open coding used in grounded theory. In total, we identified six strategies and six challenges for REpresentational State Transfer (REST) and event-driven communication via message brokers. The strategies mainly focus on API backward compatibility, versioning, and close collaboration between teams. The challenges include change impact analysis efforts, ineffective communication of changes, and consumer reliance on outdated versions, leading to API design degradation. We defined two important problems in microservice API evolution resulting from the challenges and their coping strategies: tight organizational coupling and consumer lock-in. To mitigate these two problems, we propose automating the change impact analysis and investigating effective communication of changes as open research directions.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

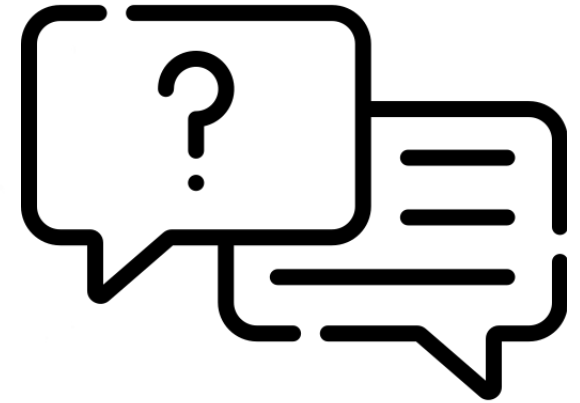
Research Questions

RQ1: What means do developers use to exchange messages between services, and how do they document them?

RQ2: Which strategies do developers follow to introduce and communicate API changes in loosely coupled systems?

RQ3: Which challenges do developers face when introducing and communicating API changes in loosely coupled systems?

Methodology



- Semi-structured interviews
- n=17 interview participants
- Sampled from 11 companies

RQ2: Strategies

RQ2: Strategies – Backward compatibility

API evolution strategy	# P	# C
Stay compatible and avoid unexpected breaking changes	17	11
Work around breaking changes	17	11
Regression test the API	10	8
Think ahead and design a dynamic API	6	6

Participants working with MSA and large systems apply regression testing

RQ2: Strategies – Versioning

API evolution strategy	# P	# C
Version the API	17	11
Create a new version on breaking changes	17	11
Expose multiple versions simultaneously	13	8

Architects recommend exposing all versions within one service

RQ2: Strategies – Close Collaboration

API evolution strategy	# P	# C
Collaborate with other teams	15	9
Actively involve consumer teams	14	8
Follow the API-first approach	11	8

Architects prefer meetings, senior developers prefer API previews

RQ3: Challenges

RQ3: Challenges – Change Impact Analysis

API evolution challenge	# P	# C
Manual change impact analysis is error-prone	14	11
Code changes affect the API unexpectedly	9	7
Understanding consumed APIs' changes is effort	9	7

Participants recommend close collaboration to mitigate this challenge

RQ3: Challenges – Consumers

API evolution challenge	# P	# C
Consumers rely on API compatibility	12	7
Communication with other teams lacks clarity	9	7
Consumers might be unknown	7	5
Informal communication channels	17	11
Communication suffers from hierarchy	6	4

Generally applicable communication strategy does not exist

RQ3: Challenges – Maintainability

API evolution challenge	# P	# C
API maintainability and usability degrade over time	14	9
Outdated API versions add maintenance overhead	10	8
Backward compatibility increases technical debt	9	6

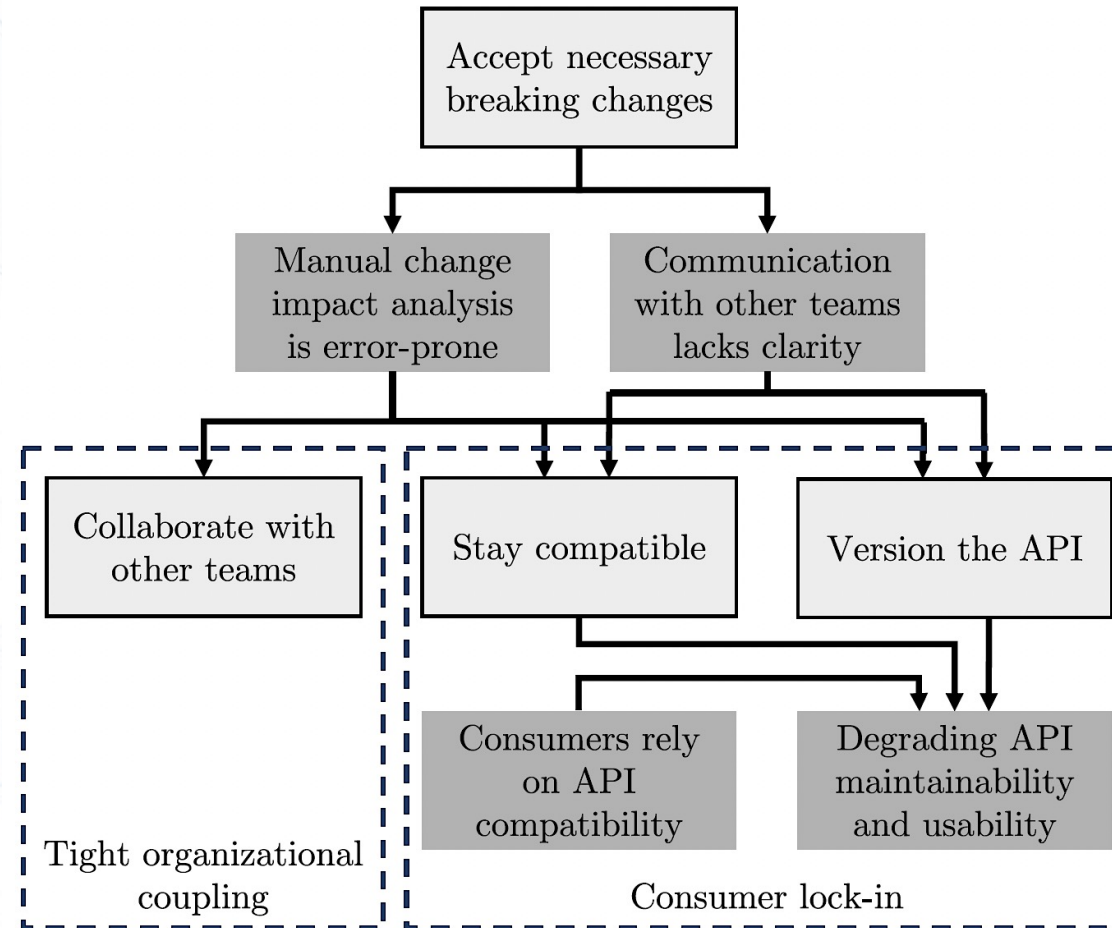
Most architects and senior developers acknowledge this challenge

Identified underlying problems

Identified underlying problems

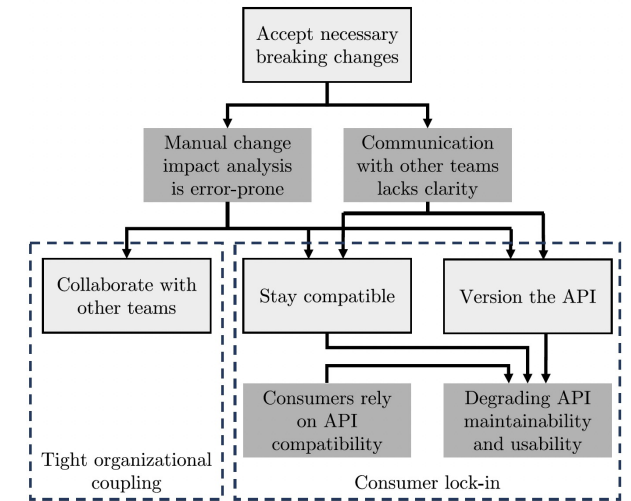
Strategy

Challenge



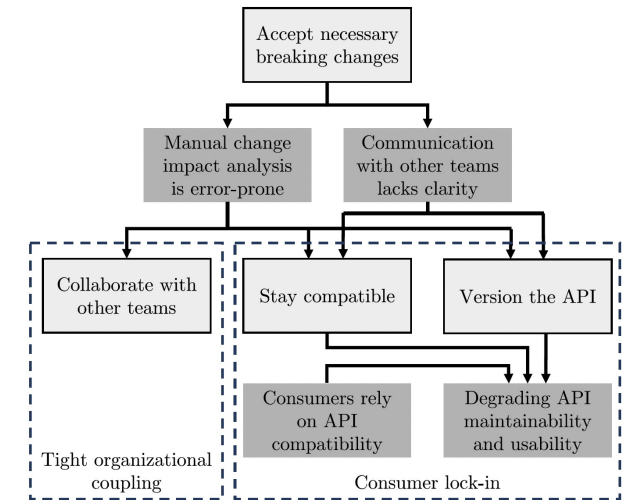
Tight organizational coupling

- Replaces the manual change impact analysis
- Requires frequent meetings and collaboration
- Leads to fragmented and undocumented knowledge within teams
- *“You see, the whole topic is really organizational-heavy, organizational and planning-heavy. [...] The technical part is then just doing it”*

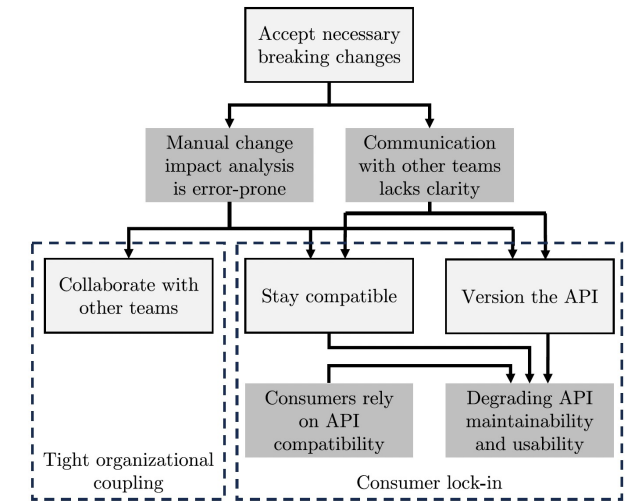


Consumer lock-in

- Replaces the communication with other teams
- Requires continuous support for outdated API versions
- Degrades the API design and increases technical debt
- *“The cost of these workarounds that we do, I don't know. I don't dare to estimate that.”*

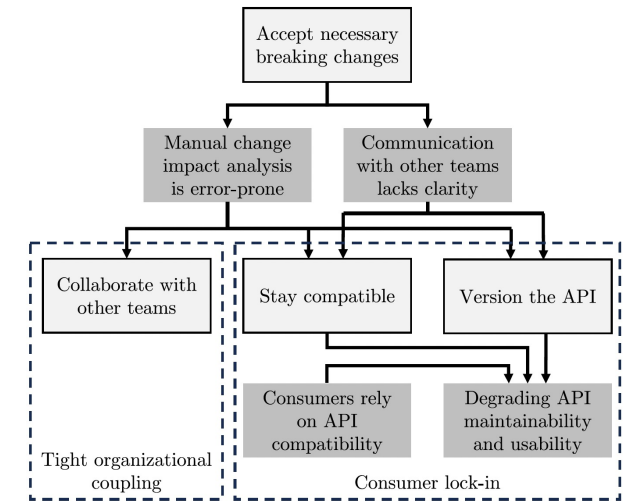


Open Research Directions



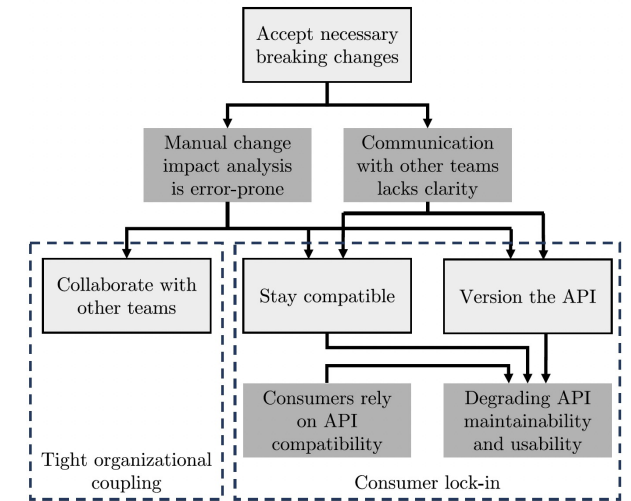
Open Research Directions

- Automating the change impact analysis
 - Impact of the provider's source code changes on the API
 - Impact of the API changes on individual consumers



Open Research Directions

- Automating the change impact analysis
 - Impact of the provider's source code changes on the API
 - Impact of the API changes on individual consumers
- Providing effective change communication
 - Reliably notifying
 - affected consumers
 - with customized change logs





Microservice API Evolution in Practice:
A Study on Strategies and Challenges

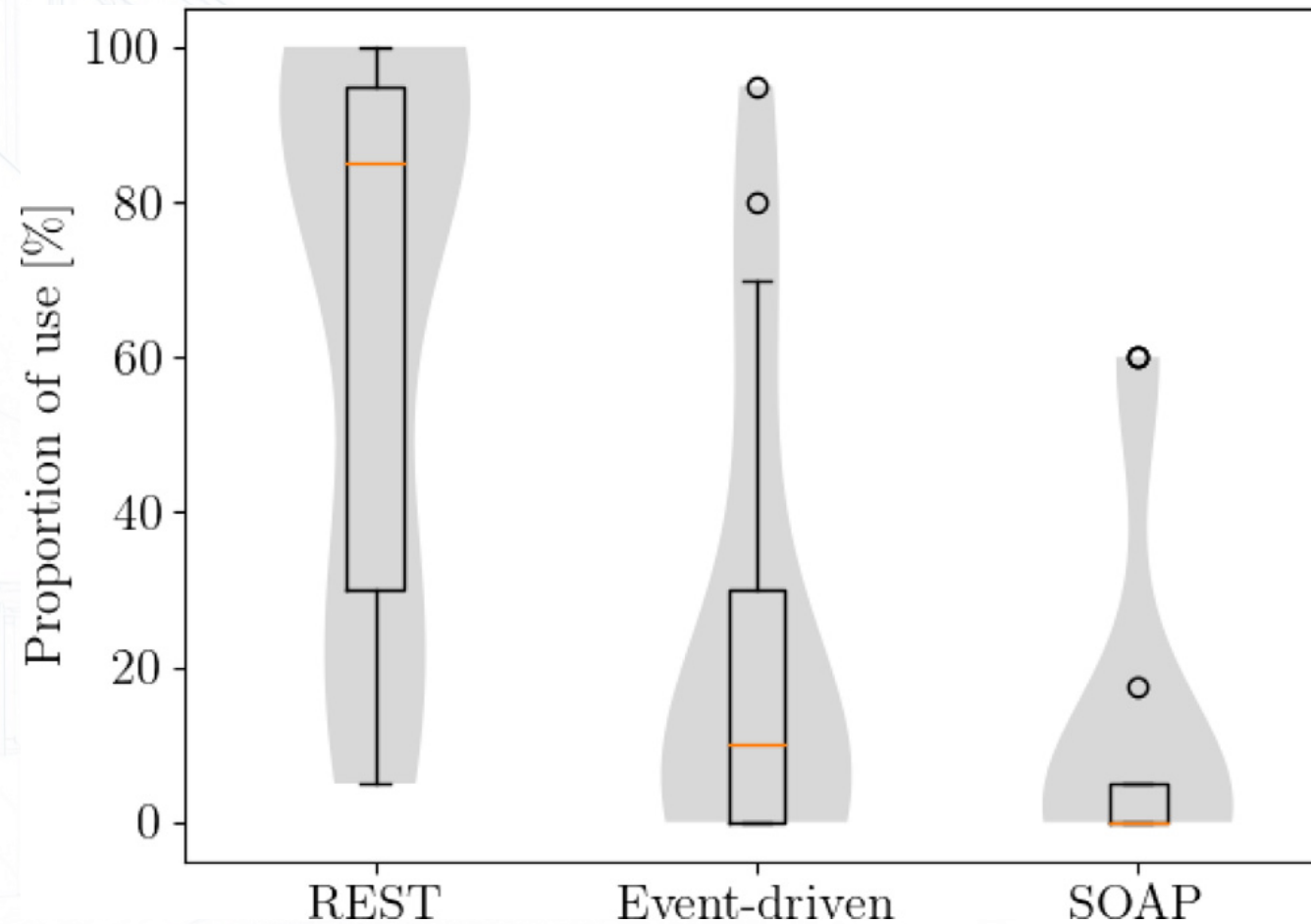
Interview Participants

Participant code	Highest education	Technical role	Exp ^a (yrs)	Exp ^b (yrs)	System architecture	System size ^c	Team size
C1-P1	Bachelor	Developer	7	3	MSA and monolith	25	6
C2-P1	Ph.D.	Principal architect	13	6	MSA	–	–
C2-P2	Ph.D.	Architect	10	6	MSA	50	6
C2-P3	Master	Architect	10	6	MSA	50	6
C3-P1	Master	Architect	10	3	Services	6	10
C3-P2	Bachelor	Developer	4	4	Services	7	10
C3-P3	Technical high school	Developer	7	5	Services	20	17
C5-P1	Master	Senior developer/Technical lead	10	4	Services	30	5
C5-P2	Technical high school	Senior developer	7	6	Services	30	5
C6-P1	Bachelor	Senior developer/Technical lead	15	3	MSA	15	15
C7-P1	Master	Developer	4	3	Services and monolith	3	6
C8-P1	Bachelor	Developer	6	3	MSA	10	10
C9-P1	Ph.D.	Developer	2	1	MSA and Function-as-a-Service	70	3
C10-P1	Master	Architect	14	7	MSA and monolith	15	7
C11-P1	Master	Architect	9	7	Self-contained systems and monolith	20	6
C11-P2	Master	Principal architect/Department head	20	7	Self-contained systems and monolith	–	–
C12-P1	Technical high school	Senior developer/Product manager	25	5	Services	40	13

Interview Participants

Company code	Industry field	Company size	# P
C1	Construction	Large	1
C2	Access management	Large	3
C3	Automotive	Large	3
C5	Video processing	Medium-sized	2
C6	Retail	Large	1
C7	Monitoring	Large	1
C8	Process digitization	Small	1
C9	E-commerce	Small	1
C10	Traffic management	Large	1
C11	Research and higher education	Large	2
C12	E-mobility	Large	1

RQ1: Message Exchange Techniques



RQ2: More Strategies

API evolution strategy	# P	# C
Accept necessary breaking changes	17	11
Understand the reasons for breaking changes	17	11
Consider structural and behavioral changes	5	4
Internally, just break (and fix) it	11	10
Abstract external systems' APIs	6	5

RQ3: More Challenges

API evolution challenge	# P	# C
Governmental services are uncooperative	6	4
Event-driven communication evolution is disregarded	7	4