9 CRITERIA TO ASSESS YOUR PROJECT'S MATURITY & IMPROVE

Petyo Dimitrov

21 May 2024

About me

17 years in Software Engineering

8 years as a Software Architect

Head of Architecture at Qinshift Bulgaria (ex. Musala Soft)

PhD in Computer Science from Technical University Sofia & Guest lecturer

Agenda

What is technical maturity?

How to assess it?

What are the 9 criteria?

How to organize the results?

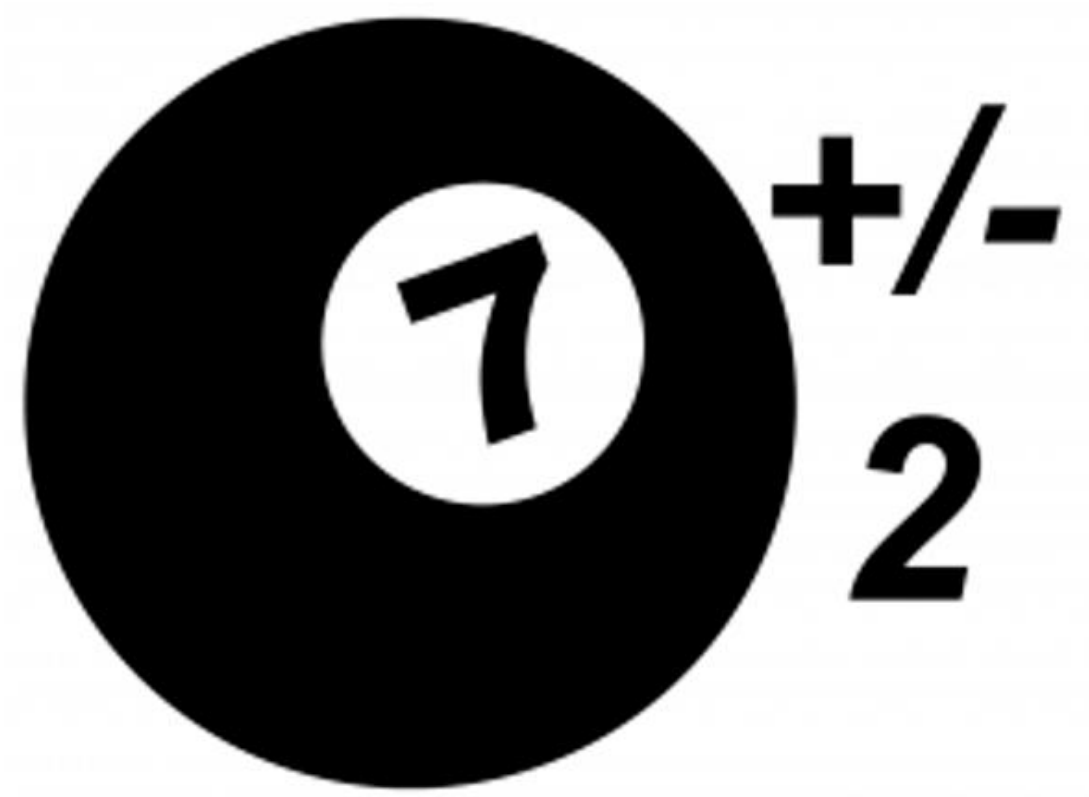
How to communicate them?

What is (technical) maturity?

maturity is realizing that
Tom was not the villain

Why does it matter?

Why 9 criteria?



Miller's Law

Security

Code quality

Deployability

Testability

Operability

Integration

Cloud capability

Processes

3rd-party

Methodology – Kick-off meeting(s)

- Explain the process & its importance
- Clarify the scope of the assessment (get client sign off)
- Identify key criteria depending on the project
- Align with the project's timeline
- Request documents & other materials (e.g. source code, access)
- Send reference materials for the criteria
- Ensure the participation of key team members (schedule meetings with specific attendees and agenda)

Methodology – Review provided materials

- **Documents*** – requirements spec, project plan, test plan, HLA, infrastructure info (IaC), API docs, development process, user guide, etc.
- **Board**
- **Automated code review** – via Sonar, NDepend, Dependency Checker, etc.
- **Selective manual code review(s)**
- **Exploratory testing** – check key functionality, pen testing

Methodology – Populate checklist

reference

Criterion	Assessment	Description
1. Integration & API Design	satisfactory	Does the application offer high-quality interfaces for integration into the system
1.1 API provides access to important functions and data	fulfilled	
1.2 Is the API based on a future-proof standard such as HTTP + REST, HTTP + GraphQL, Message Broker (with AMQP, Kafka Protocol, MQTT, STOMP), gRPC	fulfilled	
1.3 API is easy to understand (read) and comprehensible	not fulfilled	
1.4 API documentation is available (could be manual)	fulfilled	
1.5 Asynchronous integration is utilized (i.e., events)	not fulfilled	
1.6 Automatically generated API documentation is available	not fulfilled	
1.7 Non-functional requirements for key APIs are defined and followed (performance, availability, security, versioning, error handling, etc.)	not fulfilled	
Total fulfilled	3	***Evaluation basis:*** *excellent: 6+ criteria met | good: 4+ criteria met | satisfactory: 2+ criteria met*

Security (1/9)

System hardening

Error & logon failures logging

1FA

Logging of user-IDs, system time and type of change

Logging of user-IDs, system time, and change before & after

Validation of input data

Encryption in transit

Usage of secure hashes

2FA

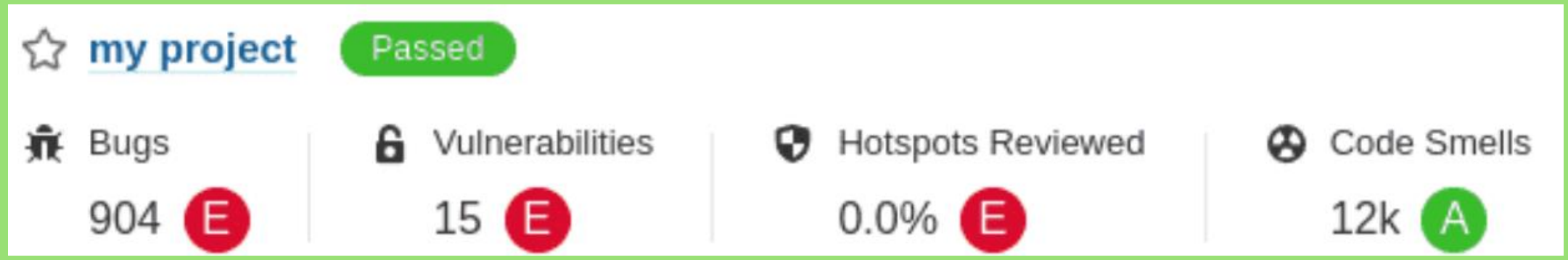
Encryption at rest

Security (1/9)

- Address OWASP Top 10 (injection, XSS, outdated deps., etc.)
- **Static Application Security Testing** (SAST) is used (e.g. Sonar)
- **Penetration Testing** is performed/planned (via ZAP, SSL Labs, etc.)
- **Dependency Vulnerabilities** are addressed periodically
- **Principle of Least Privilege** access control (for Cloud, 3rd parties, etc.)

Code quality (2/9)

- Automated **Static Code Analysis** (SCA) - via Sonar*
- **Technical Debt** as % of Backlog
- **Coding standards** are defined
- **Peer reviews** are conducted
- **Pair programming**



Deployability (3/9)

- **CI/CD pipelines** (+ as Code)
- Independent build & deployment
- Versioned **Artifact repository**
- **Rollback mechanism**
- **Environment parity**

Testability (4/9)

- High test **Coverage**
- Efficient **Test pyramid** distribution
- Test code as **production code**
- **Performance tests**
- Execute as **part of CI/CD**
- **Test Driven Development**

Operability (5/9)

- **Standardized logging**
- **Central configuration** & **user administration**
- **Health-checks** & metrics
- **Backup & restore** procedures
- **Root Cause Analysis** is performed
- **Automated restart & reconnect**
- **Automated notifications** for production issues

Integration (6/9)

- API based on **REST / GraphQL / gRPC / messaging** is provided
- **API documentation** is available (+ generated automatically)
- API is easy to understand
- Asynchronous integration is used
- Non-functional requirements are defined (performance, availability, etc.)

Cloud capability (7/9)

- Runs on **virtualized hardware**
- **Containerized & orchestrated**
- Twelve Factor App principles
- **Horizontal scalability** (+ **Elasticity**)
- **Fault-tolerant design**
- **Disaster Recovery** is planned
- **Modularization** (via MSA, EDA, SOA, etc.)

Processes (8/9)

- **Project Management** and **Version Control System**(s)
- **Core project documentation** is available - glossary, architecture, environments, development process, on-boarding guide.
- **Testing strategy** is defined and documented
- **Agile ceremonies**

3rd-party dependencies (9/9)

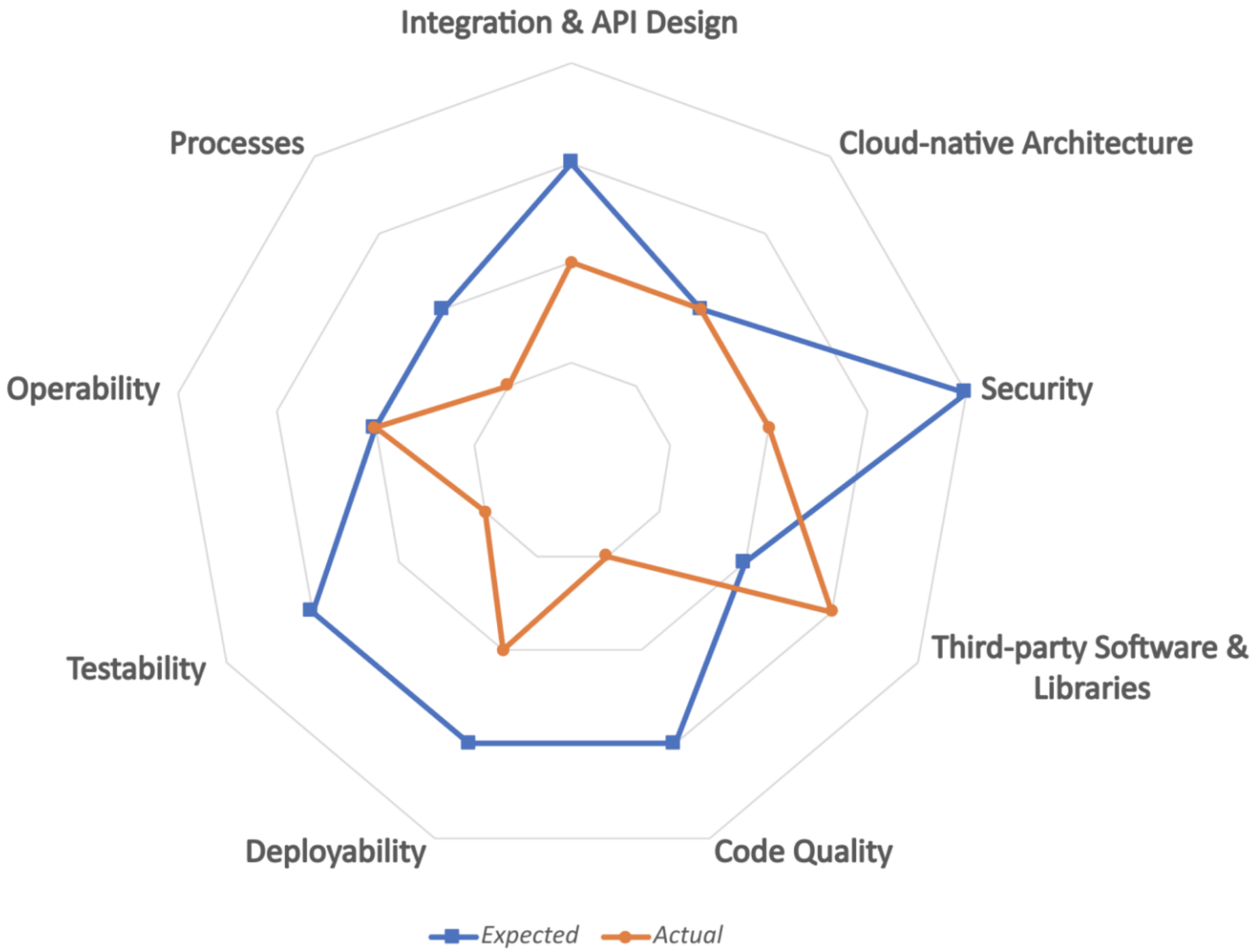
- **Up-to-date dependencies**
- Regular dependency **upgrade process**
- Dependencies with **proper OSS licenses**
- For critical components:
  - under support with SLA
  - popular & actively maintained OSS projects

Results – Template

- Serves as a professional deliverable for the client
- Gives consistency between assessments
- Simplifies onboarding of new architects

- Contents:
  - ***1-page summary*** *(for management)*
  - Project overview* (business case, architecture, technologies, infrastructure)
  - ***Recommendations*** *(for the team)*
  - Constraints (scope, access to team/client, access to materials)
  - Appendix (methodology explanation, information sources, additional materials & references)

1-page summary – radio chart



1-page summary – colored highlights

MM Criterion	Assessment (Expectation)	Details	Impact
Security	**Satisfactory (Excellent)**	**Positive**: Basic measures for security are taken. External penetration testing is planned. **Negative**: OWASP Top 10 measures are not fully implemented (brute force/XSS/CORS issues, sensitive data exposure).	Sensitive user information can be accessed by a competitor.
Code Quality	**Not satisfactory (Good)**	**Positive**: The main application features are working. **Negative**: SCA (Sonar) and linting (eslint) are not used. There are a lot of technical debts and multiple blocker+ issues. Lots of commented out blocks or test snippets in the code base. Hardcoded config.	Risks of system failures due to technical debts. Delayed development of new features due to maintainability.

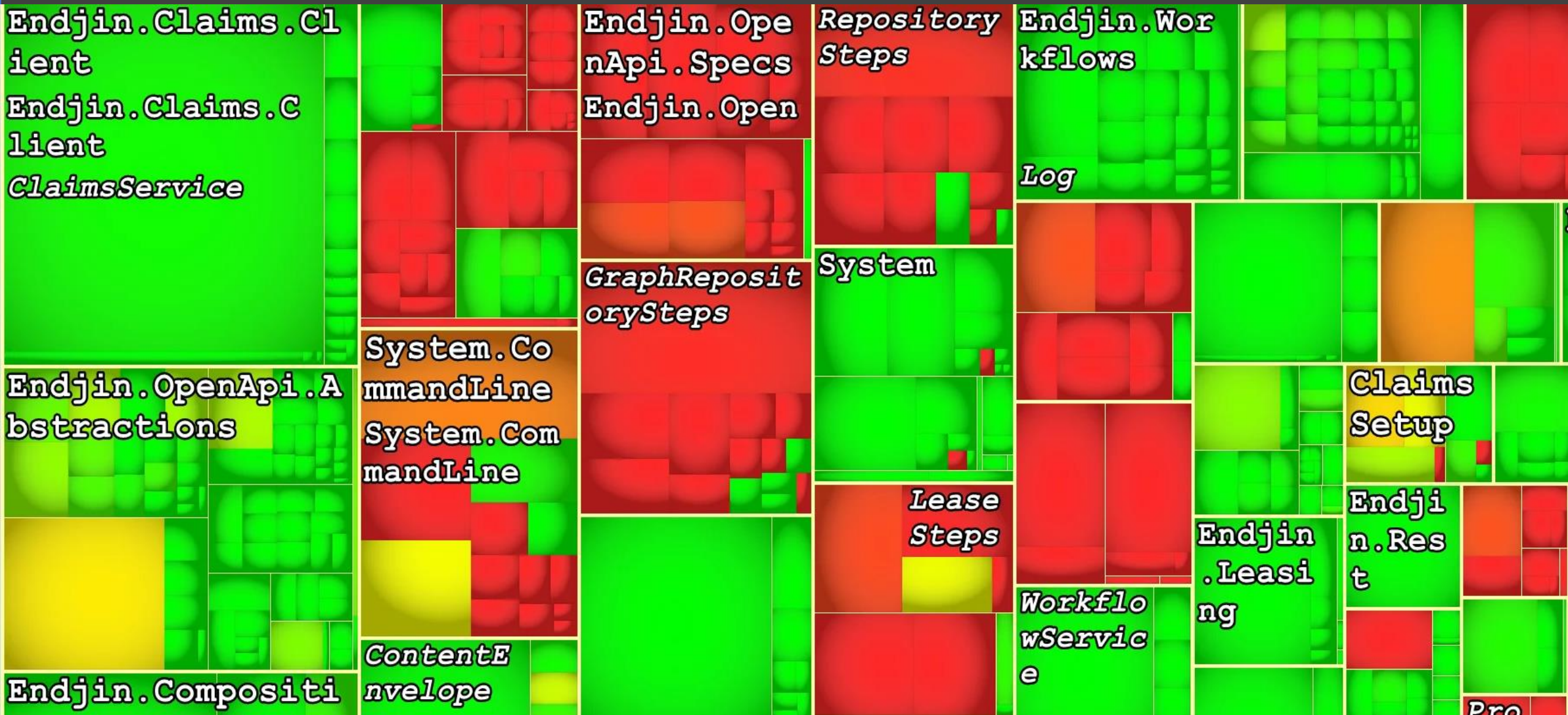
Recommendations

- Numbered, categorized, prioritized, and detailized.

Ref	Criteria	Description	Priority	Potential impact
R1	Security	Review and strengthen registration and login flow. Multiple exploits discovered - account takeover, brute forcing password/OTP due to lack of rate limit, identifying valid users based on error messages (e.g., "User not found") and responses (e.g., return user's phone number when found), weak passwords.	High	User account takeover or unauthorized access.
R2	Security	Add passwords for newly registered users in combination with OTP. Enforce some quality on the passwords, e.g., minimum 8 symbols, containing both letters and numbers.	High	2FA. Prevent sending events to external services and customers.
R3	Security	Expire OTP tokens to avoid brute force attacks.	High	Brute force attack against numerical codes.

Lessons learned – tailor to the audience

Lessons learned – gain attention via visuals



Lessons learned – engage key stakeholders early

Lessons learned – balance static & dynamic content

Lessons learned – save time for follow-ups

Conclusion

- An exhaustive and systematic approach to evaluate projects
- Allowing (somewhat) external perspective
- Targeting both business and dev key figures

Questions

Thanks!

www.linkedin.com/in/
petyo-dimitrov

petyo.dimitrov@gmail.com