Best Practices for Using the Strangler Pattern to Break Up the Monolith

Best Practices for Using the Strangler Pattern to Break Up the Monolith

Jeroen van Dun

Product Manager

Rocket Software

Herman Rensink

Distinguished Engineer

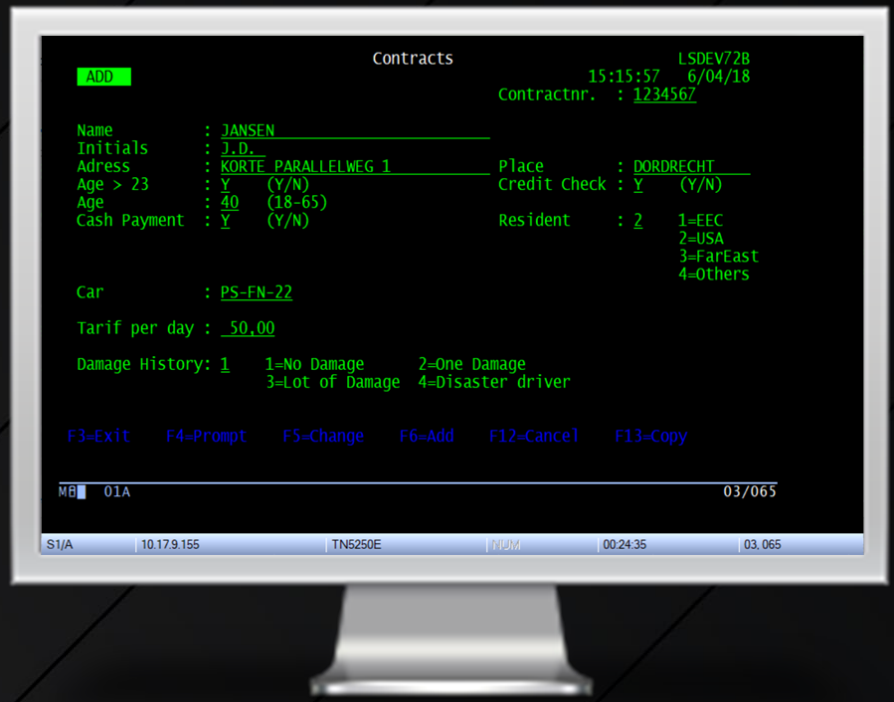
Rocket Software

Not designed for today's business

Challenging to maintain

Difficult to connect to other systems

User experience limitations



Source: Deloitte

Philosophies of Modernization

In place

Hybrid

Migrate

The Case for Modernization

IBM i is mission-critical and migration is risky!

- **Decades of business logic invested**
- **Long lead times**
- **Business disruption**
- **High costs/overruns**
- **Performant platforms**

The Case for Modernization

Businesses must evolve to stay competitive

- Modernizing IBM i applications enables businesses to better serve customers, capture new clients, and enter or even create new markets.

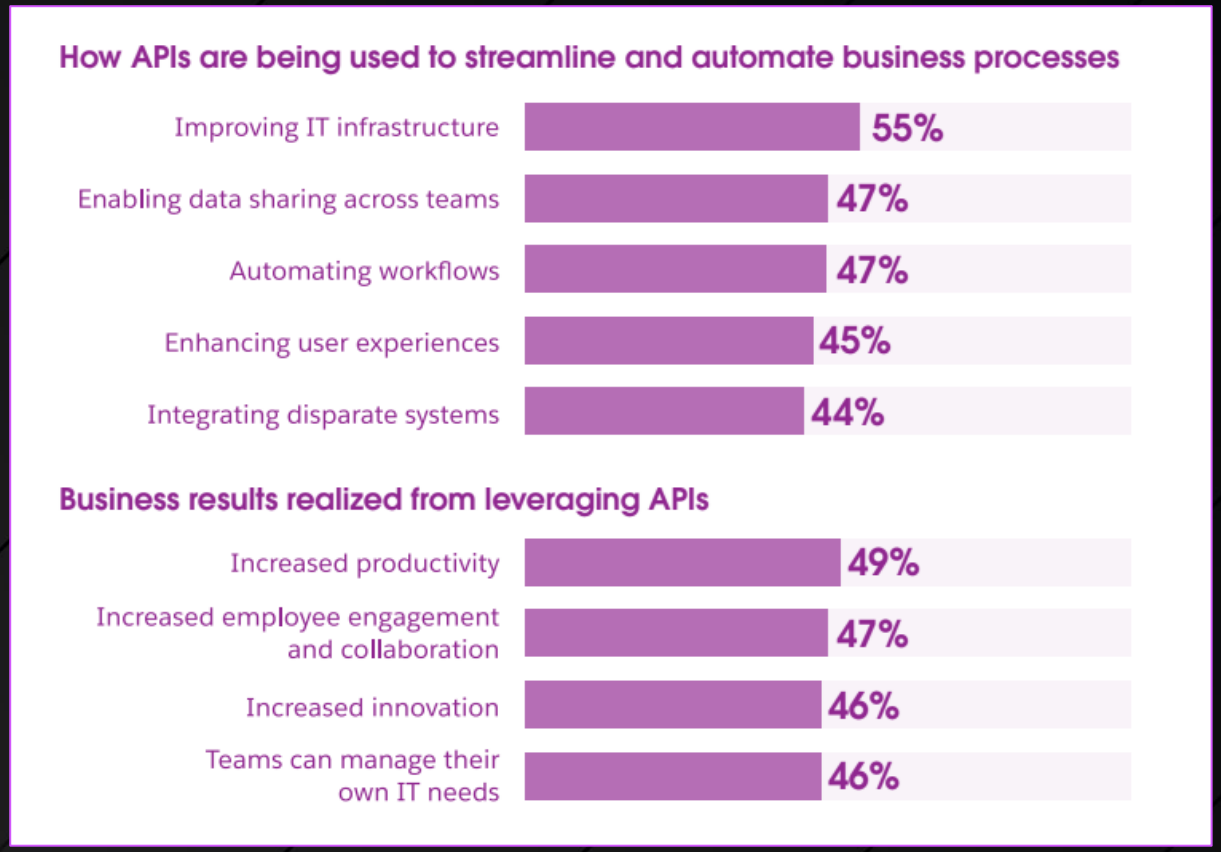
- **Modernization is the preferred path**
  - Over **70% of IBM i customers** choose modernization over migration *(IDC)*

- **Development agility is crucial**
  - **69% of IT leaders** say modernization is “very or extremely important” for business objectives over the next 12 months (Rocket Software, IT Jungle)

> *Modernizing IBM i applications enables businesses to better serve customers, capture new clients, and enter or even create new markets.*
>
> ***- ITJungle***

Role of APIs & User Experience in Modernization

- **API integration is transforming operations**
  - APIs streamline how organizations expose and consume business logic and data across platforms
- **User experience matters**
  - Modern, user-friendly interfaces drive agility, speed deployment of new features, and improve staff retention



Source: Deloitte

Philosophies of Modernization

In place

Hybrid

Migrate

The Strangler Pattern

Strangler Fig

Strangler fig is the common name for a number of tropical and subtropical plant species in the genus Ficus.

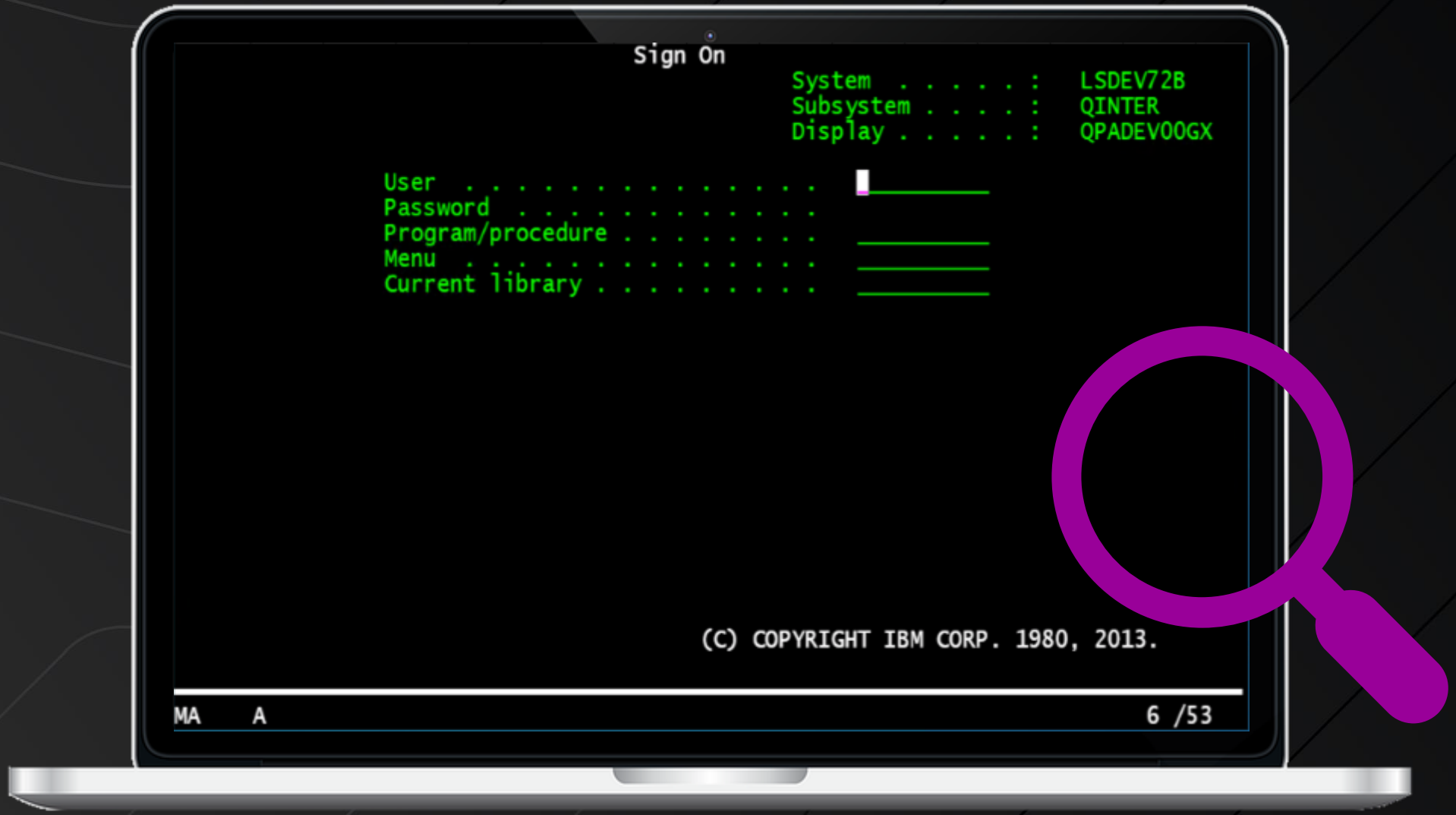
They seed in the upper branches of a tree and gradually work their way down the tree until they root in the soil.

Over many years they grow into fantastic and beautiful shapes, meanwhile strangling and killing the tree that was their host.

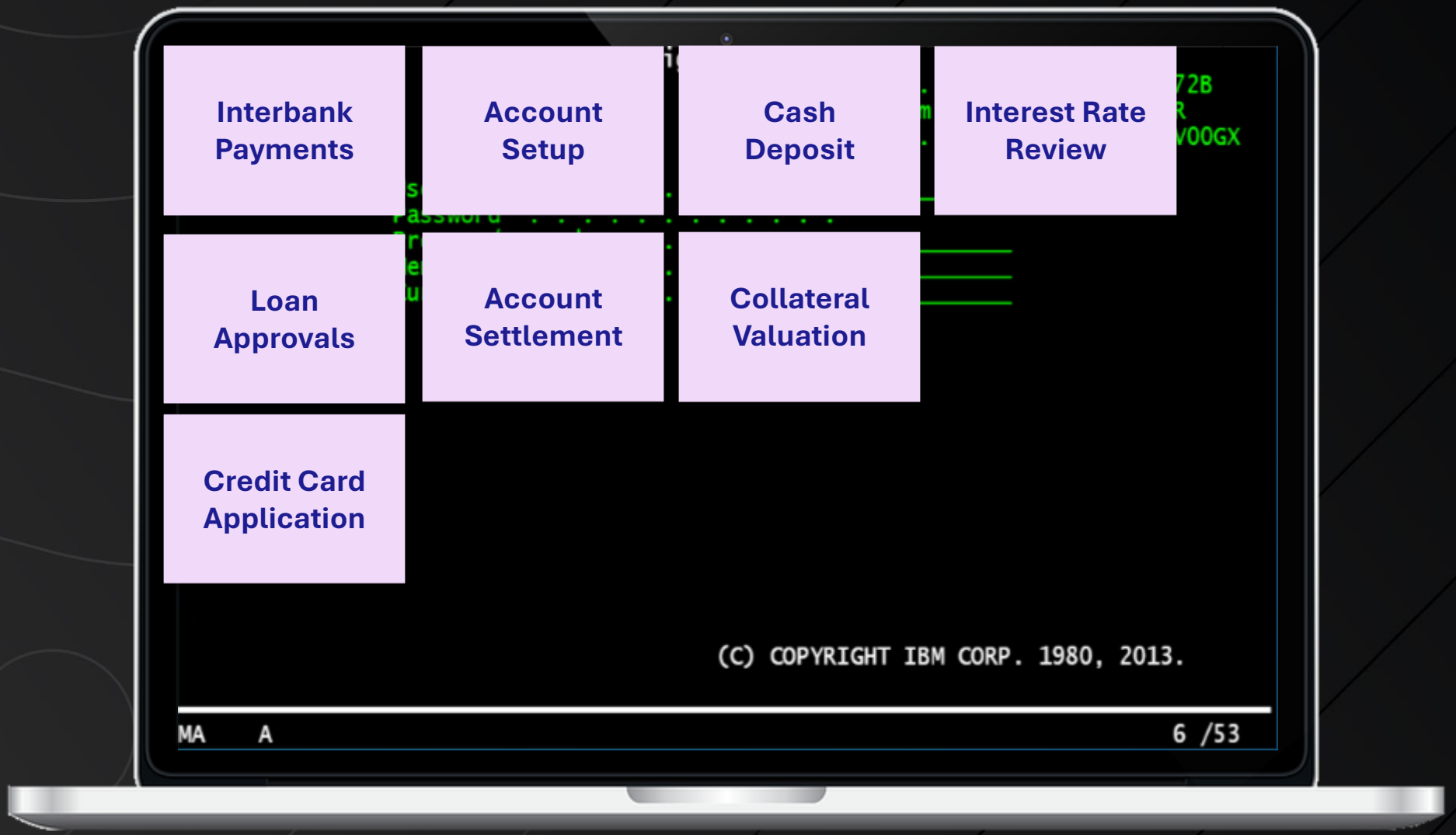
Strangler Pattern

- Examine the **business capabilities** that the **current** application provides.
- Create small strangler **services** that encapsulate the logic of each capability.
- Migrate business capabilities into **new** services.

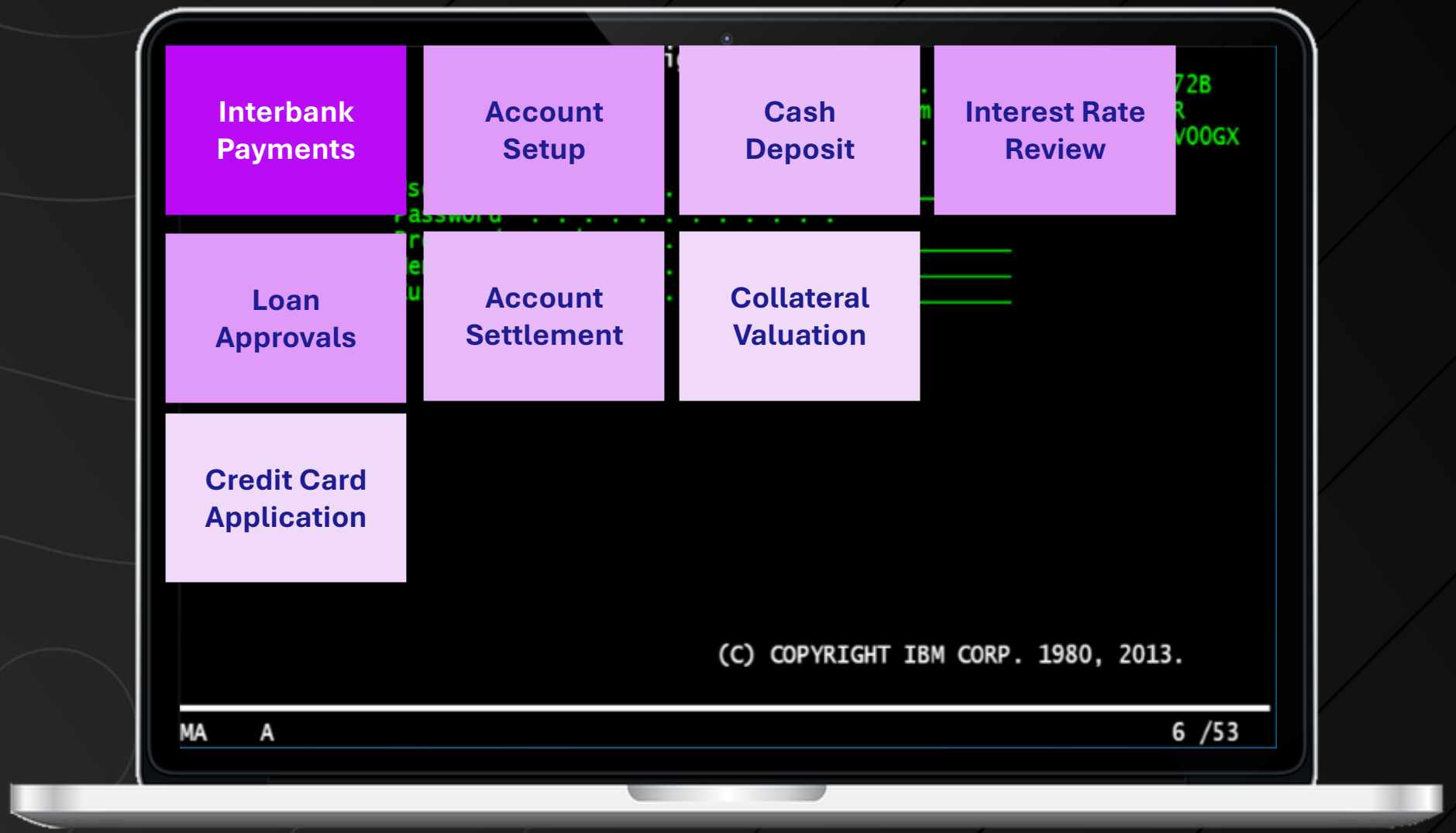
Identify the business capabilities



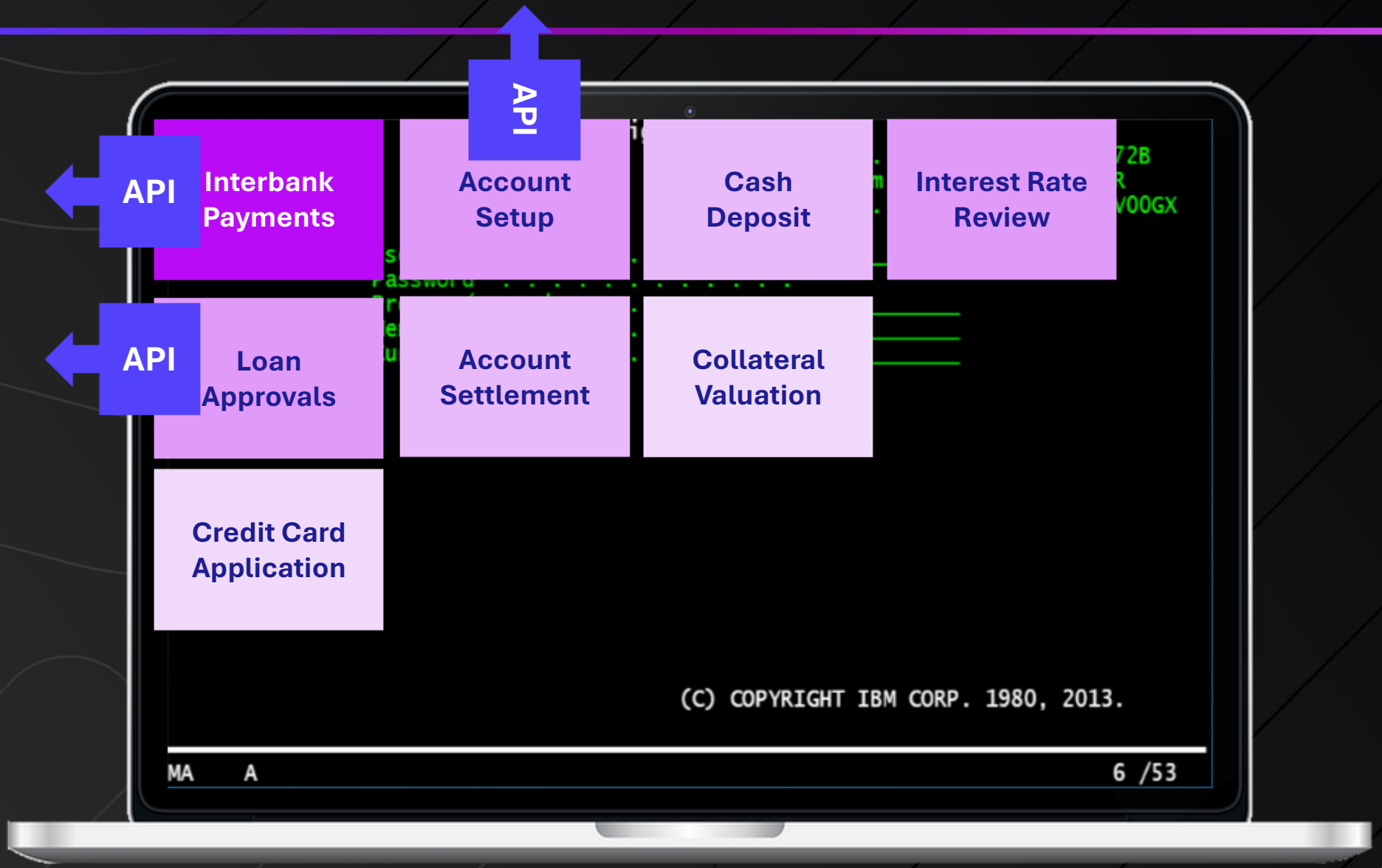
Group logic in independent bubbles...



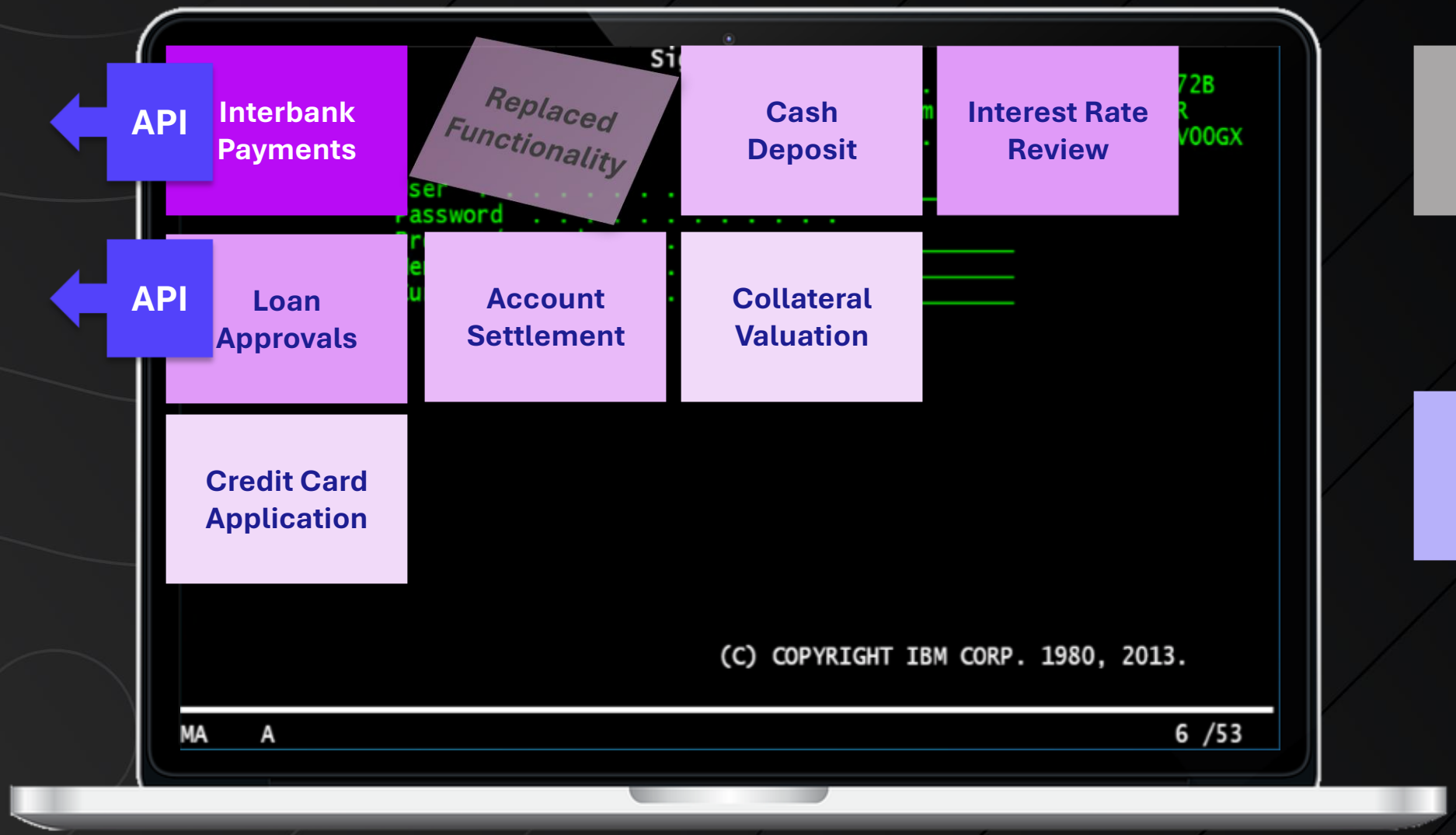
Prioritize the bubbles

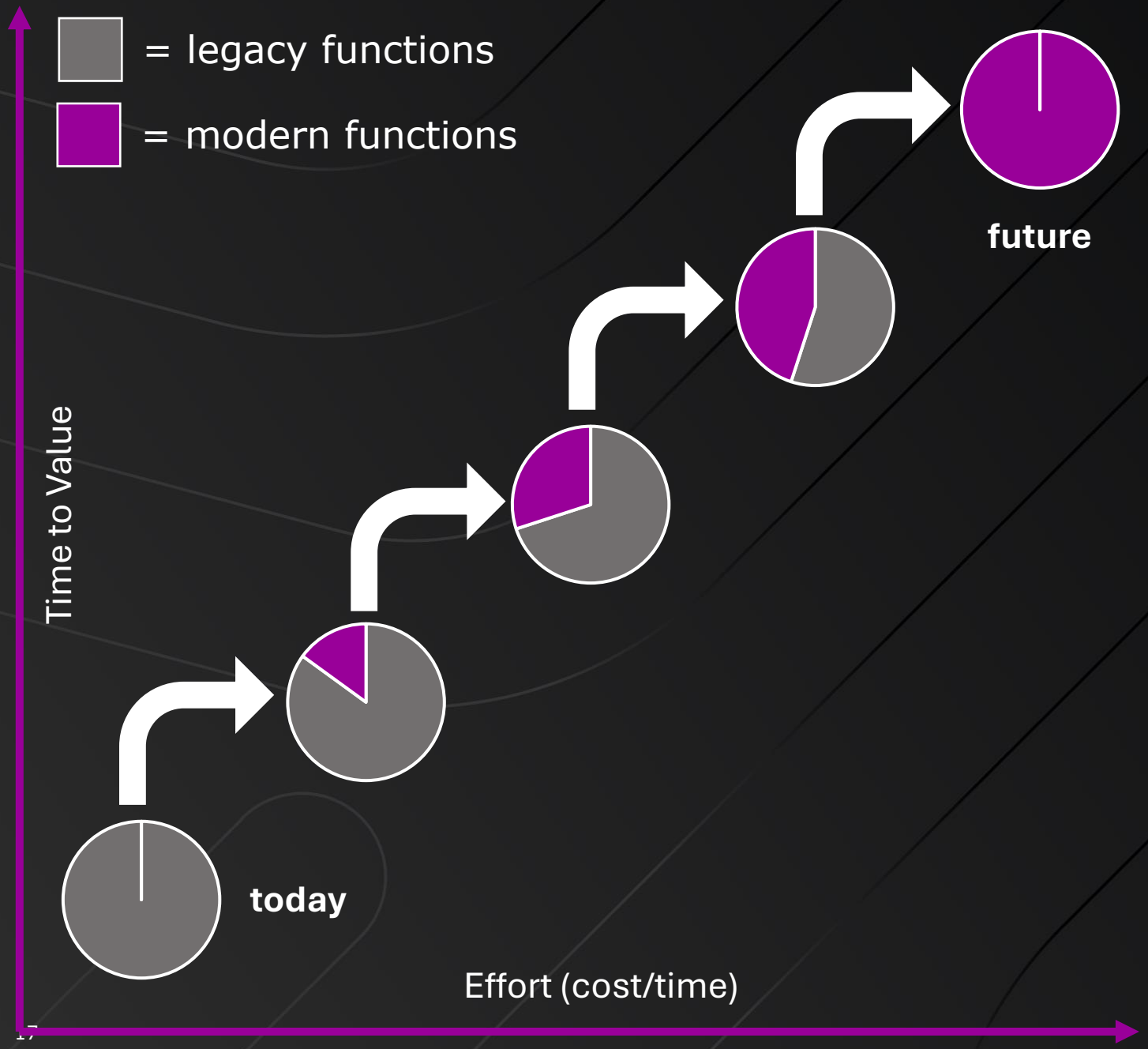


Create small strangler services for the bubbles



Chip away at the capabilities of the old application





Iterative App Modernization

Do **small** increments, avoid major rewrites.

Deliver **incremental business** value.

Breaking a huge application into smaller chunks requires **careful preparation** and a **solid testing strategy**.

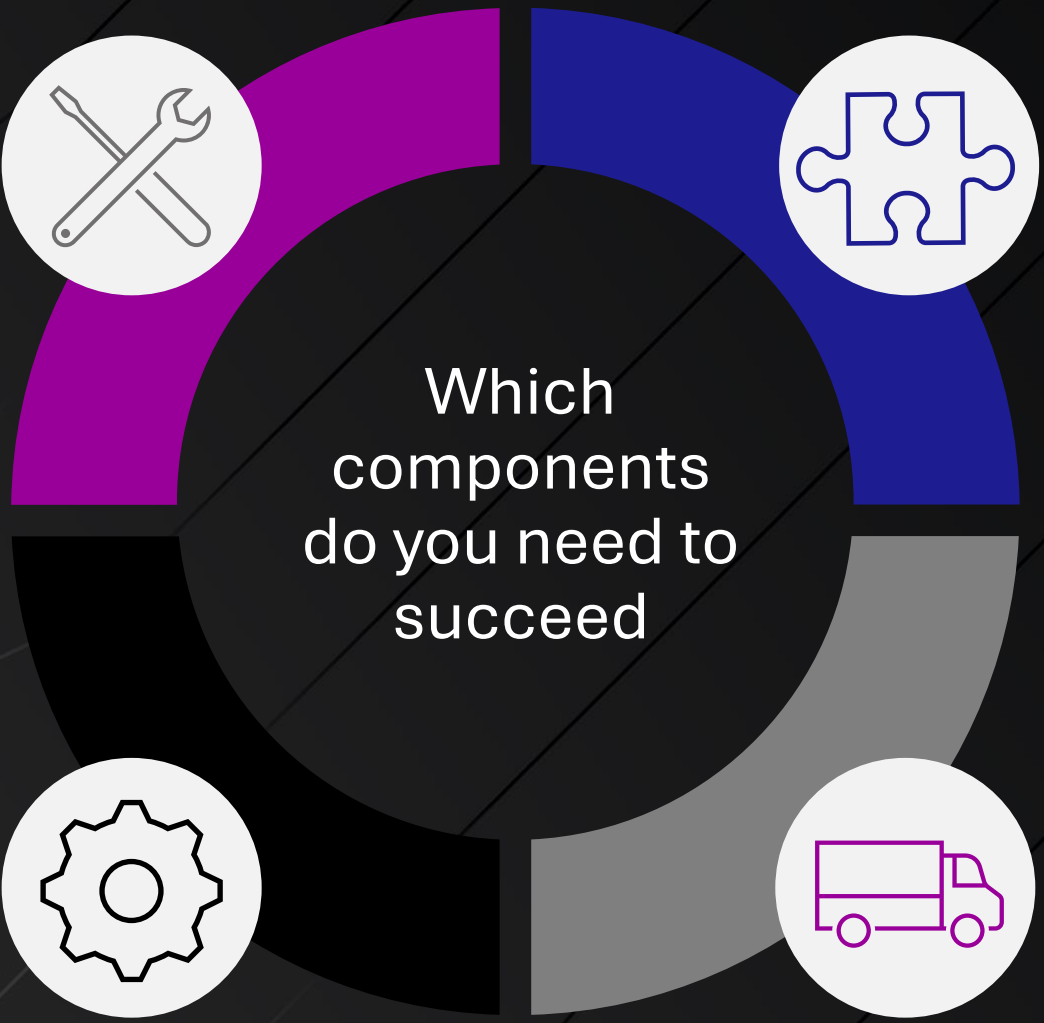
Iterative application modernization

A detailed approach

Setting the foundation for success

API management

API creation

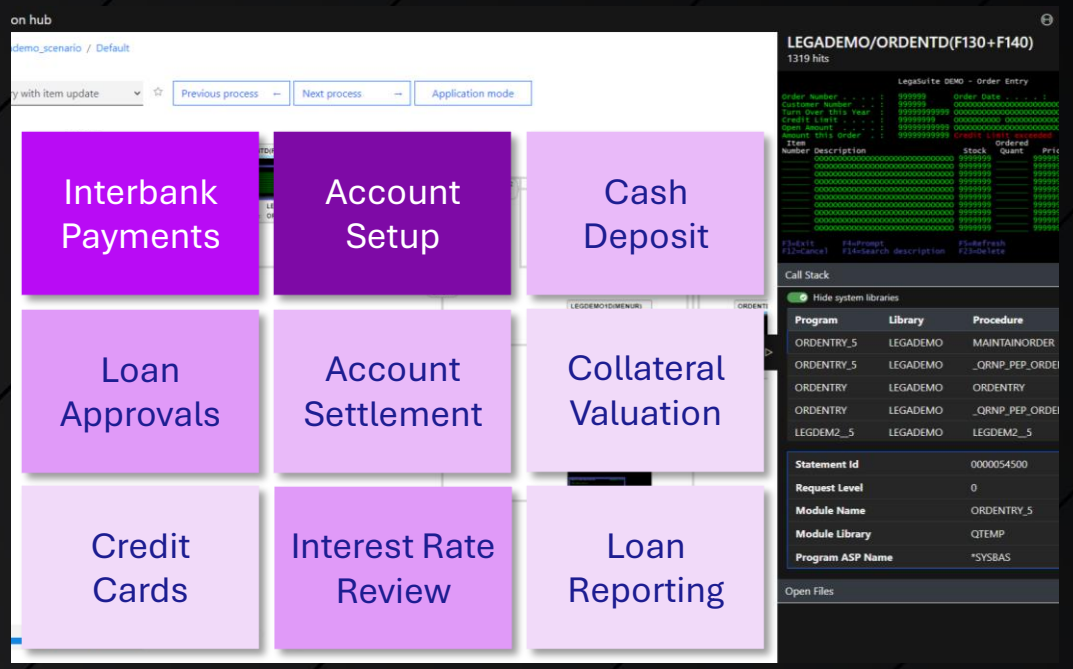


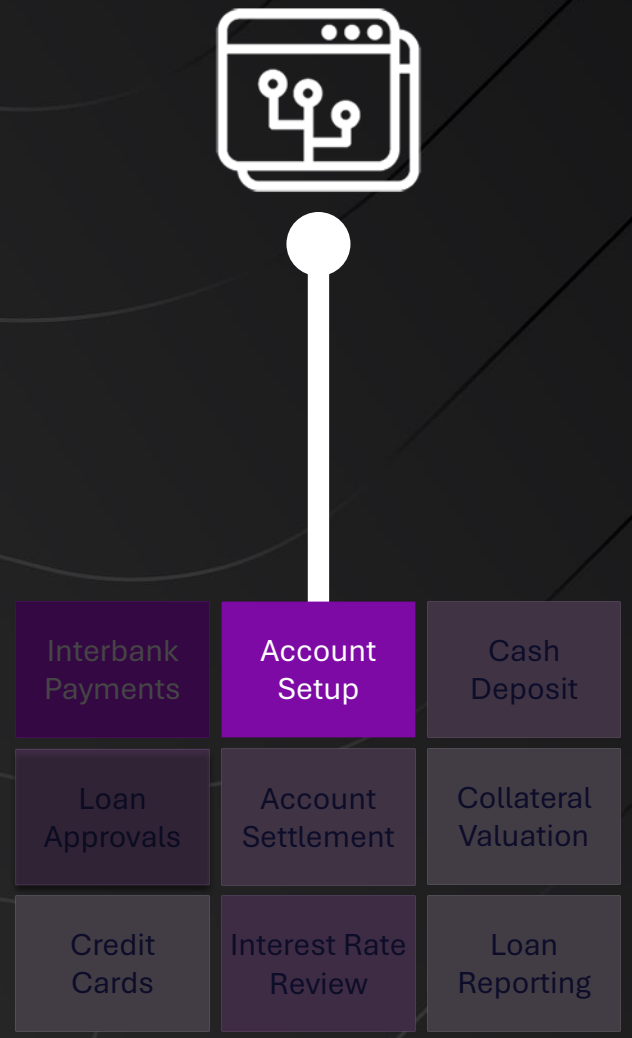
DevOps

Deployment

Collecting application insights

- How is the monolith being used?
- What business functions?
- Identify usage patterns
- Capture all user types
- Augment with source analysis



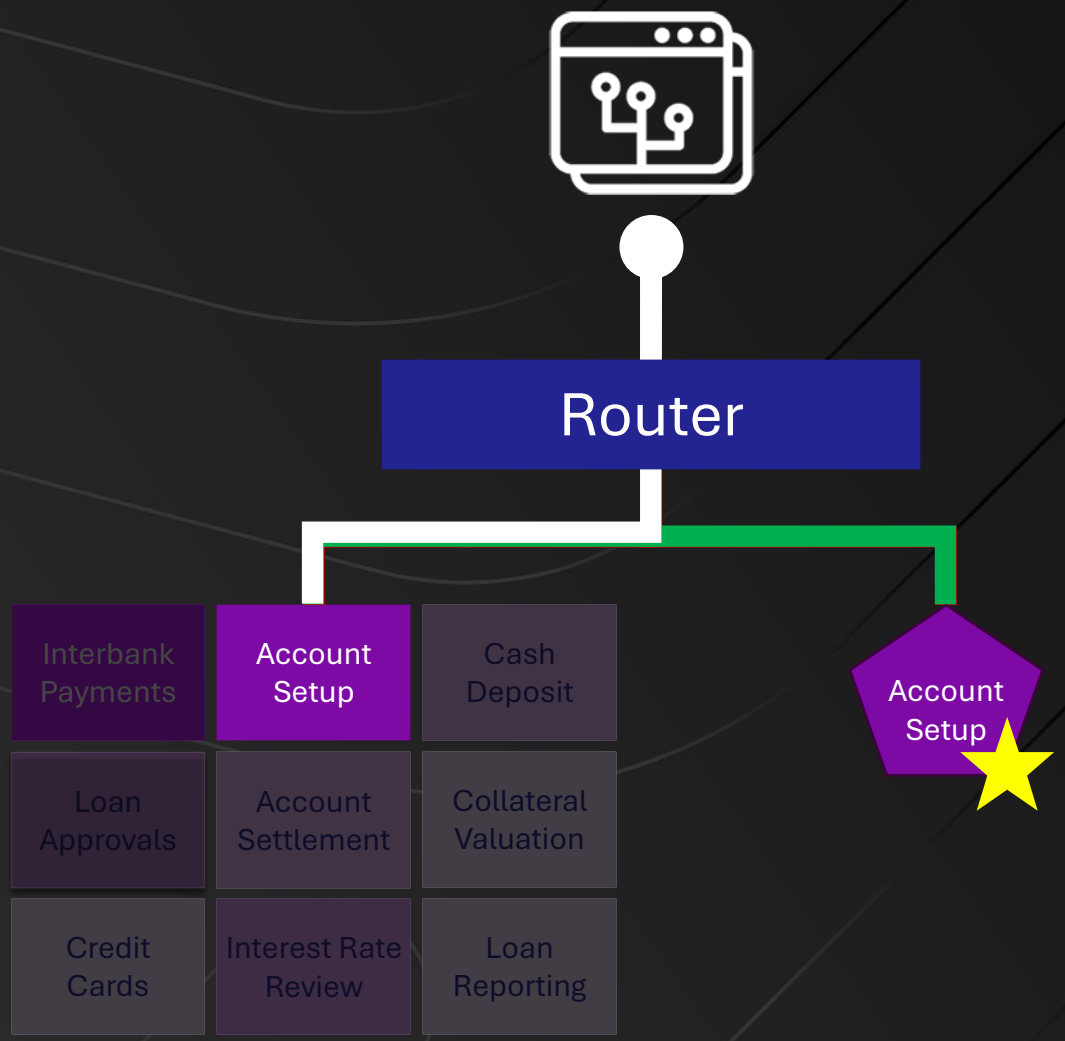


Applying the Strangler Pattern

- Analyze Application Insights
- Prioritize functions
- Turn function into a REST API

OPENAPI INITIATIVE

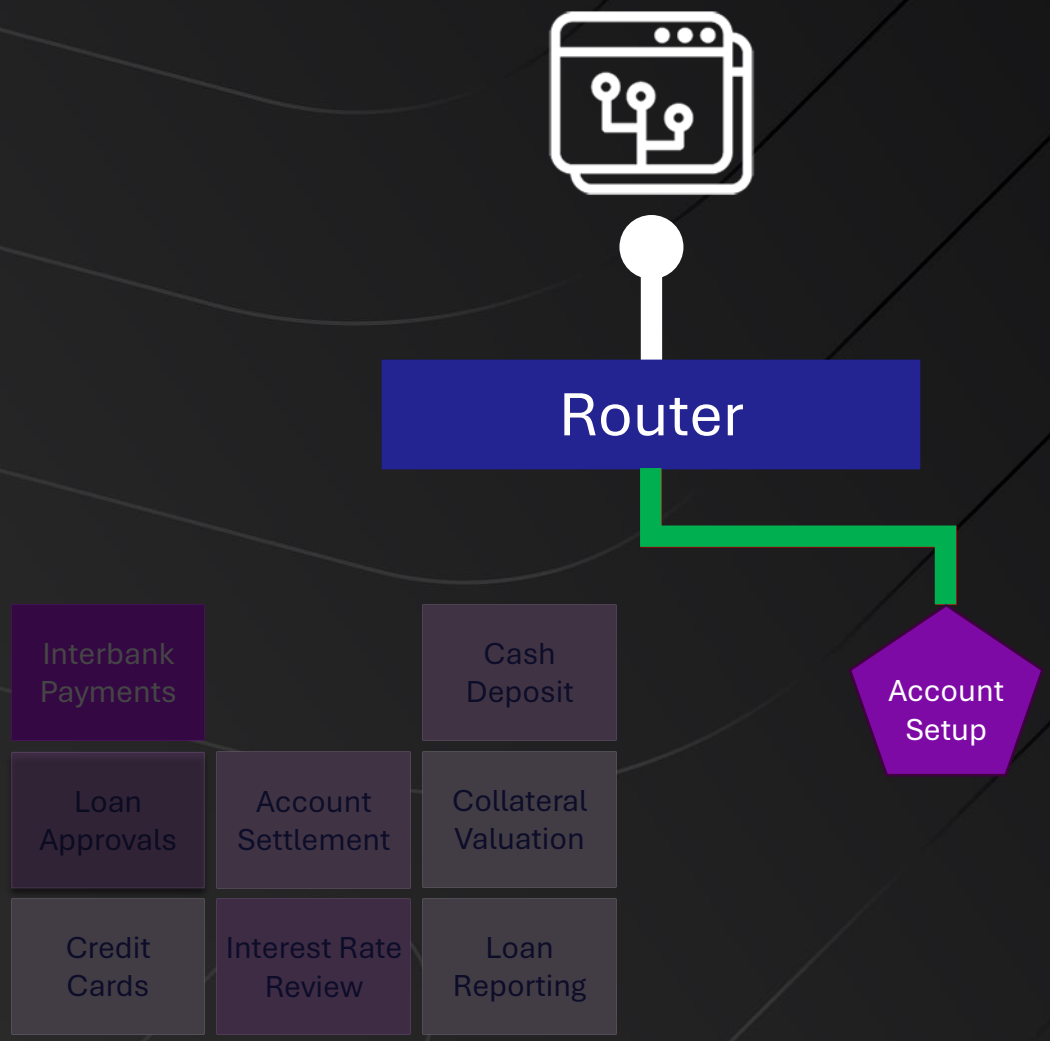
- Create a test suite for it



Applying the Strangler Pattern (2)

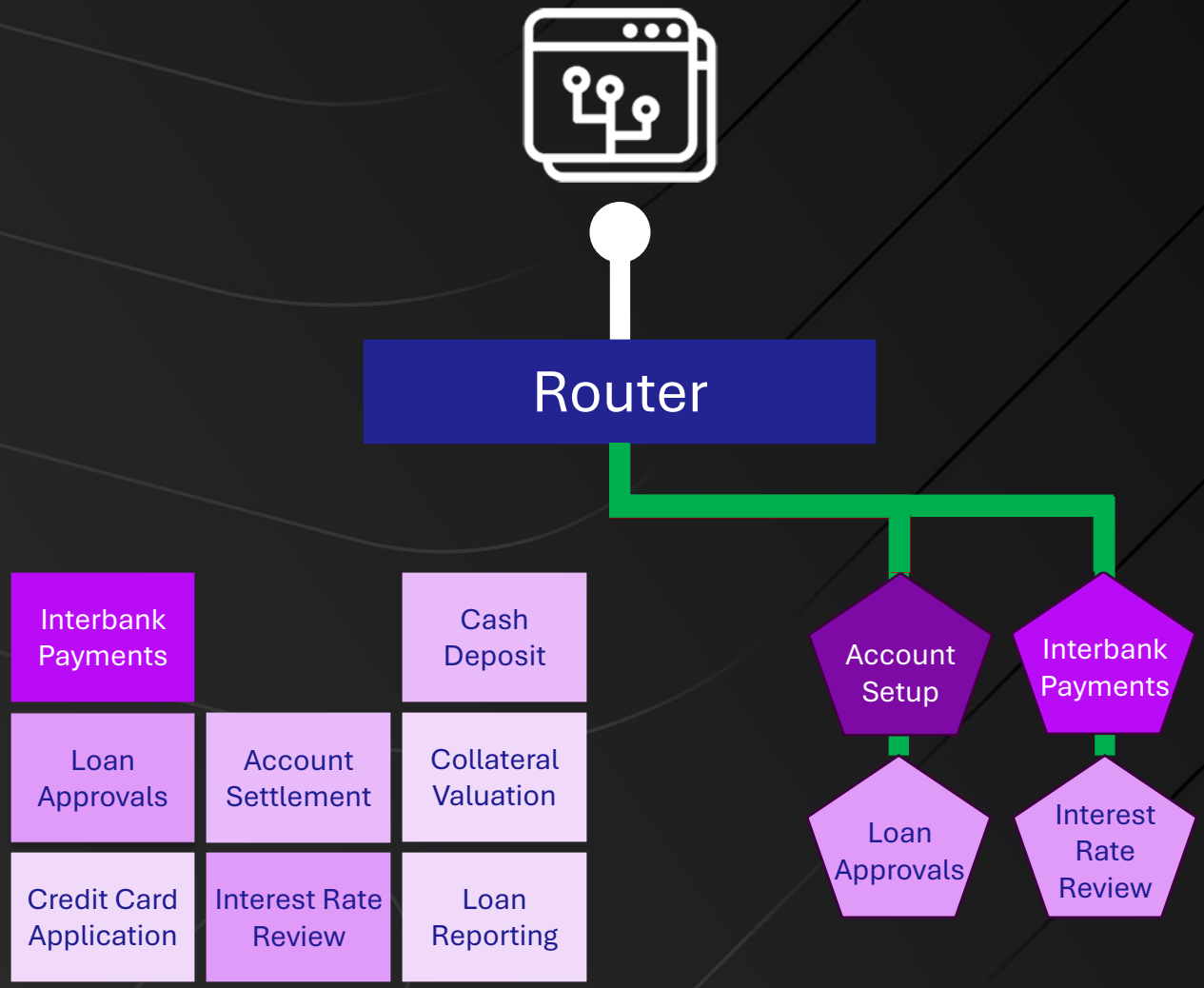
- Recreate the same function
- *same* OPENAPI INITIATIVE *spec*
- Insert router
- Switch over

★ = any language or platform



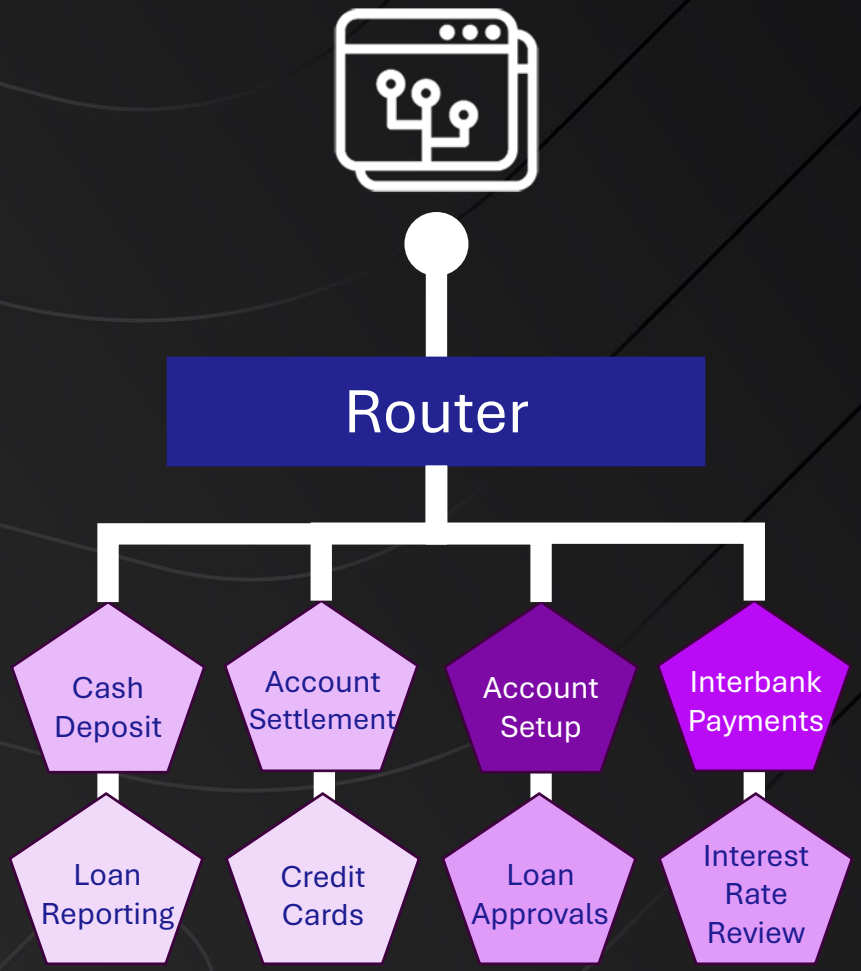
Applying the Strangler Pattern (2)

- Recreate the same function
- Same OPENAPI INITIATIVE spec
- Insert router
- Switch over
- Decommission legacy code



Applying the Strangler Pattern (2)

- Recreate the same function
- Same OPENAPI INITIATIVE spec
- Insert router
- Switch over
- Decommission legacy code
- Repeat



ACME Finance Application 1.0

10.0 OAS 3.0

Iterative Application Modernization

Account Everything about your accounts

- GET /account Get account information
- POST /account Account update
- PUT /account Add a new account.
- DELETE /account Delete an account.

Loans Everything about your loans

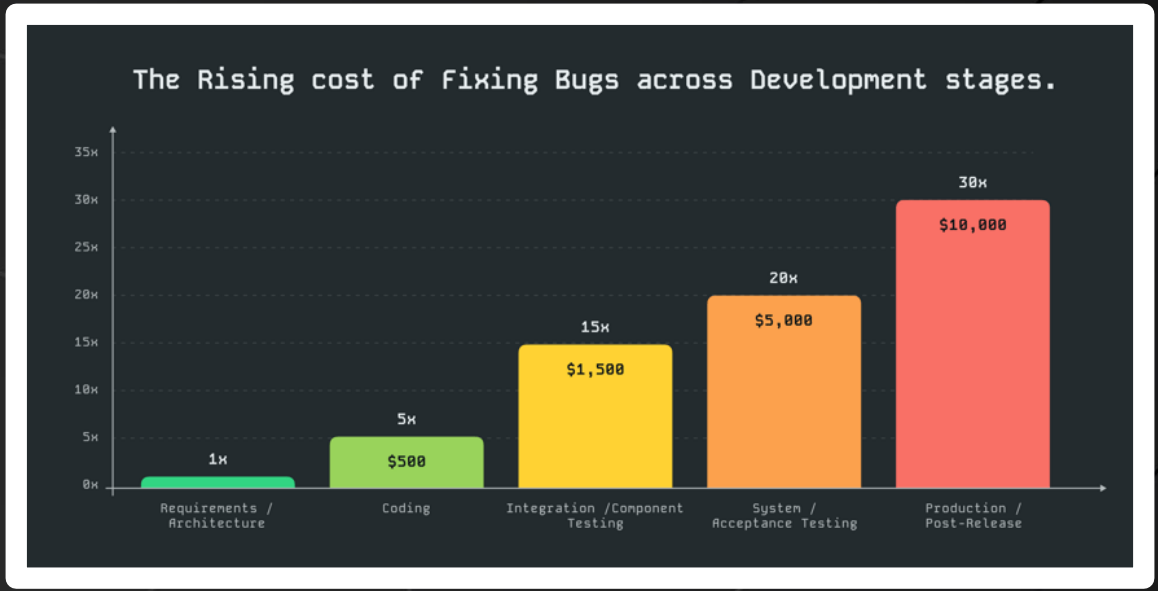
- GET /loan Get loan information
- POST /loan Loan update

Testing stages

Unit Testing	Integration Testing	System Testing	Acceptance Testing
Does the thing I created do what I expected it to do?	Does the thing I created work with the things others created?	Does the thing I created do what it was designed to do?	Does my end user like the thing I created? How is the performance?
Test new code with unit testing	*Make sure old and new work well together*	*End to end testing of the mix of old and new*	*Run old and new side by side*

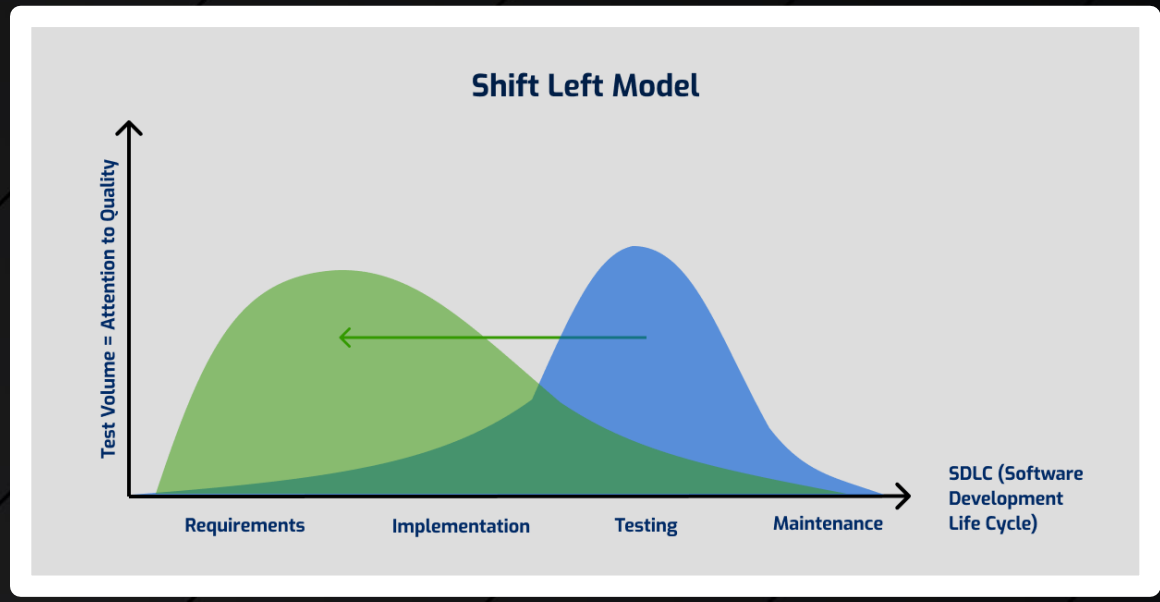
The 10x rule and Shift Left Testing

Bugs get **~10x more expensive** to fix **at each later stage** of the software development lifecycle



Barry Boehm, 1981

Larry Smith, 2001



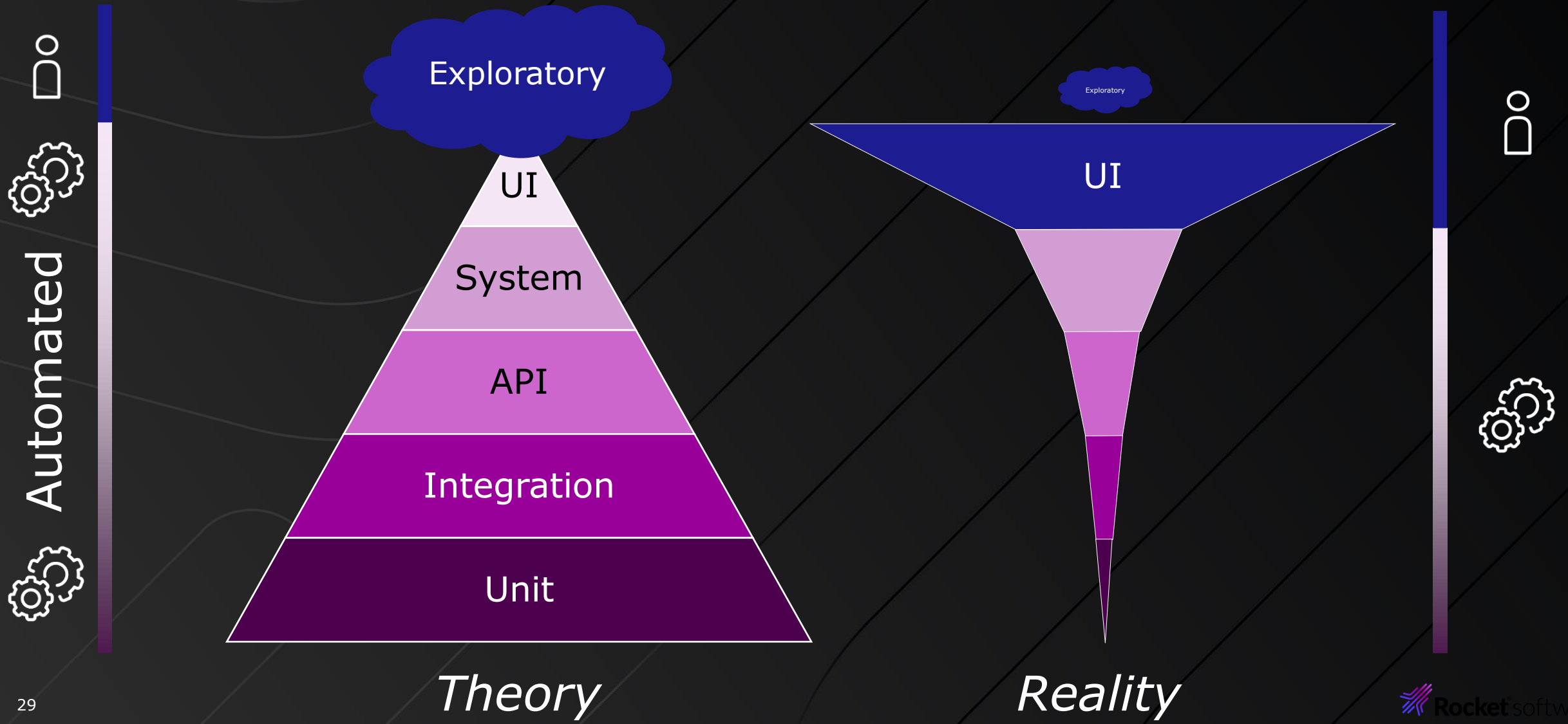
Shift Left Testing tests earlier in the SDLC

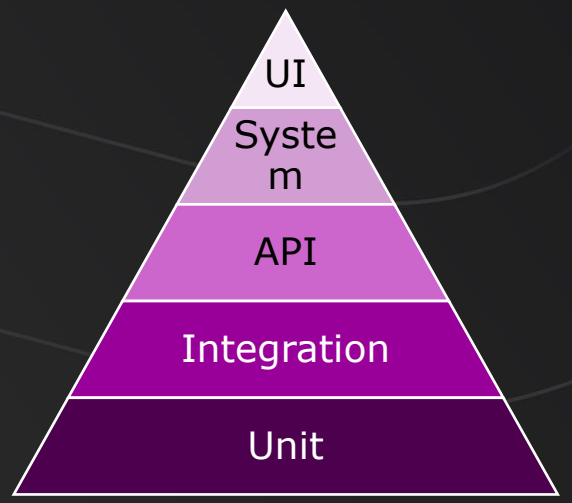
Rocket software

Test Considerations

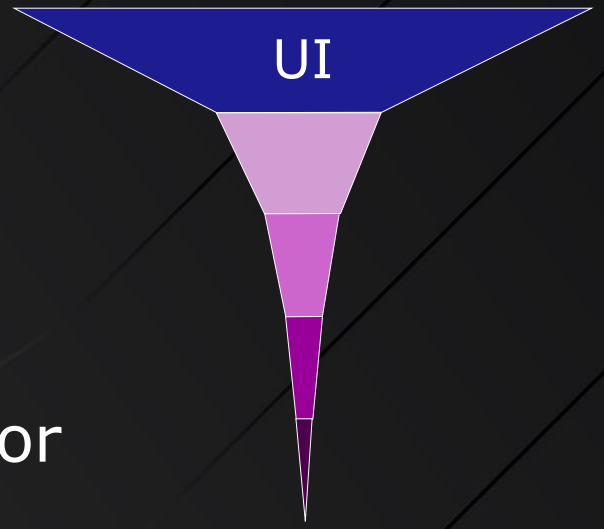
- Shift left testing
- Automate, automate, automate
- Good test data
- Generate test scripts from OpenAPI
- Measure and monitor API performance

Agile testing Pyramid *(Mike Cohn)*





or



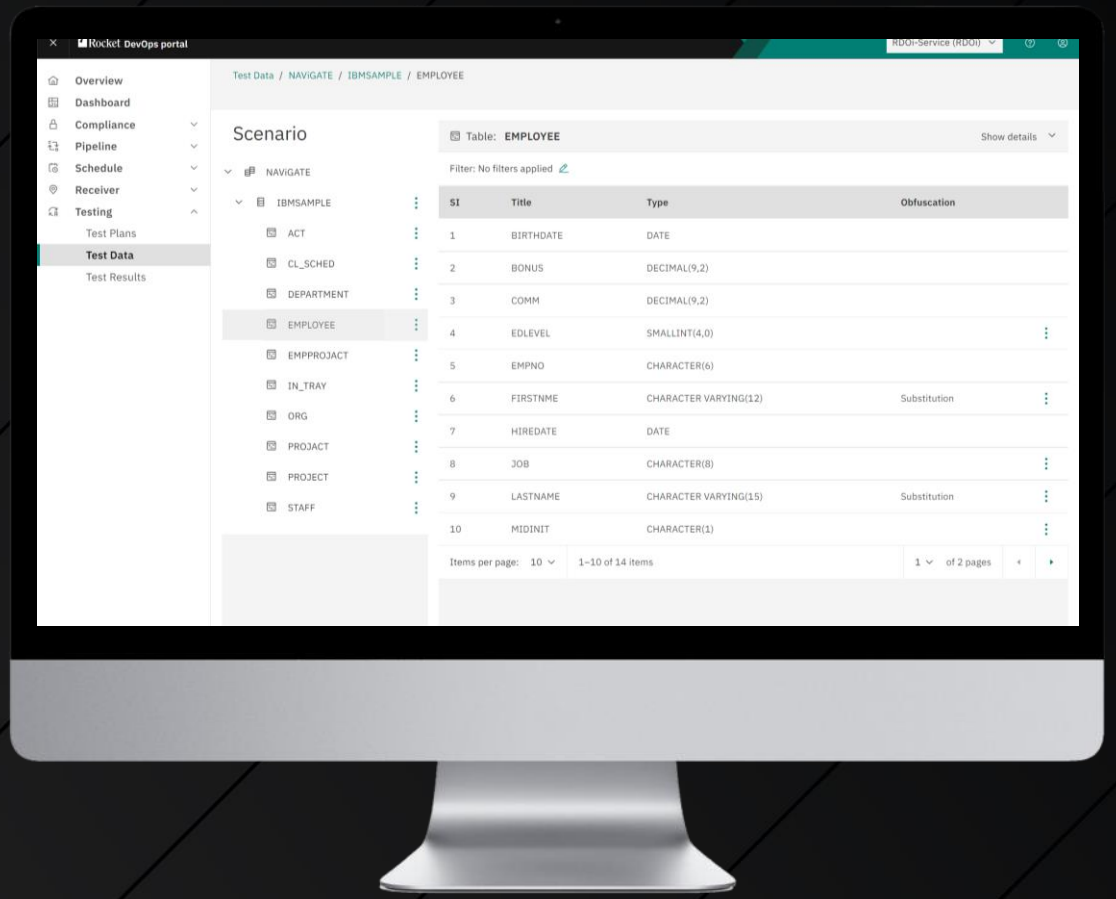
Ask the audience

What's does your pyramid look like?

- Mike Cohn's
- Cocktail glass

Test Data - Clean test data is essential

- **Test data should be:**
  - *up to date*
  - specific to the use case
  - **anonymized**
- Ensure you meet your compliance obligations
- Revert after testing

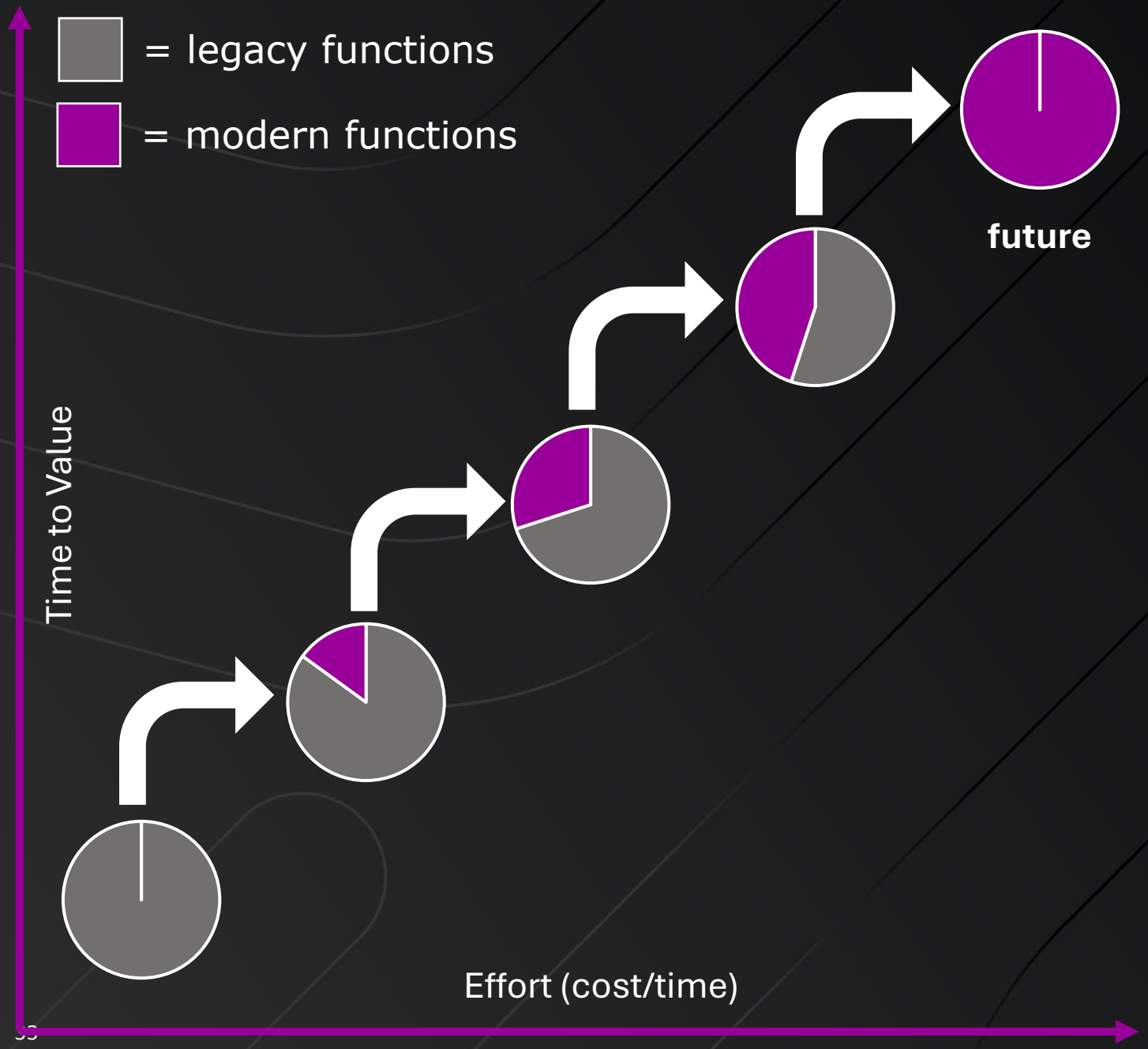


Rocket software

Ask the audience

How do you get your test data?

- We don’t
- We generate with AI
- We extract from production



Deploying the Application

Heterogeneous architecture

Where will the ***polylith*** run?

Microservice deployment should be **carefully orchestrated** and **automated**

Code deployment must be reliable

01

70% of errors in production are deployment problems – only 30% are due to faulty code (IDC)

02

Deployment is **complex** in an IBM i+ environment

03

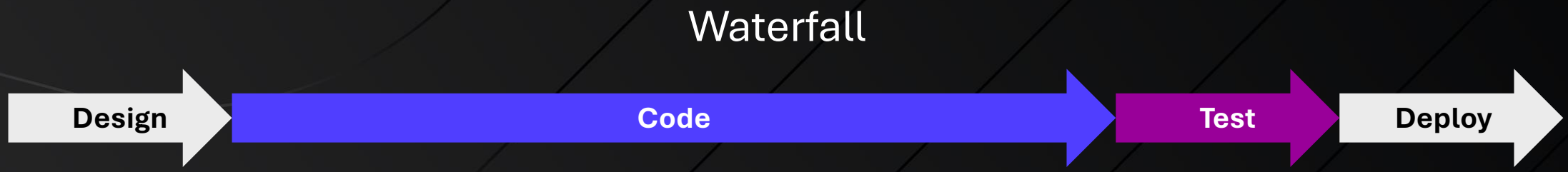
Comprehensive **control** and **visibility** of deployments is essential

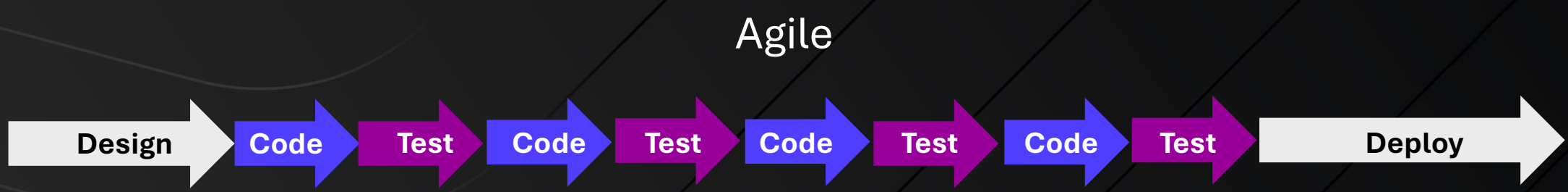
04

Testcases must cover not just software but also **deployment**

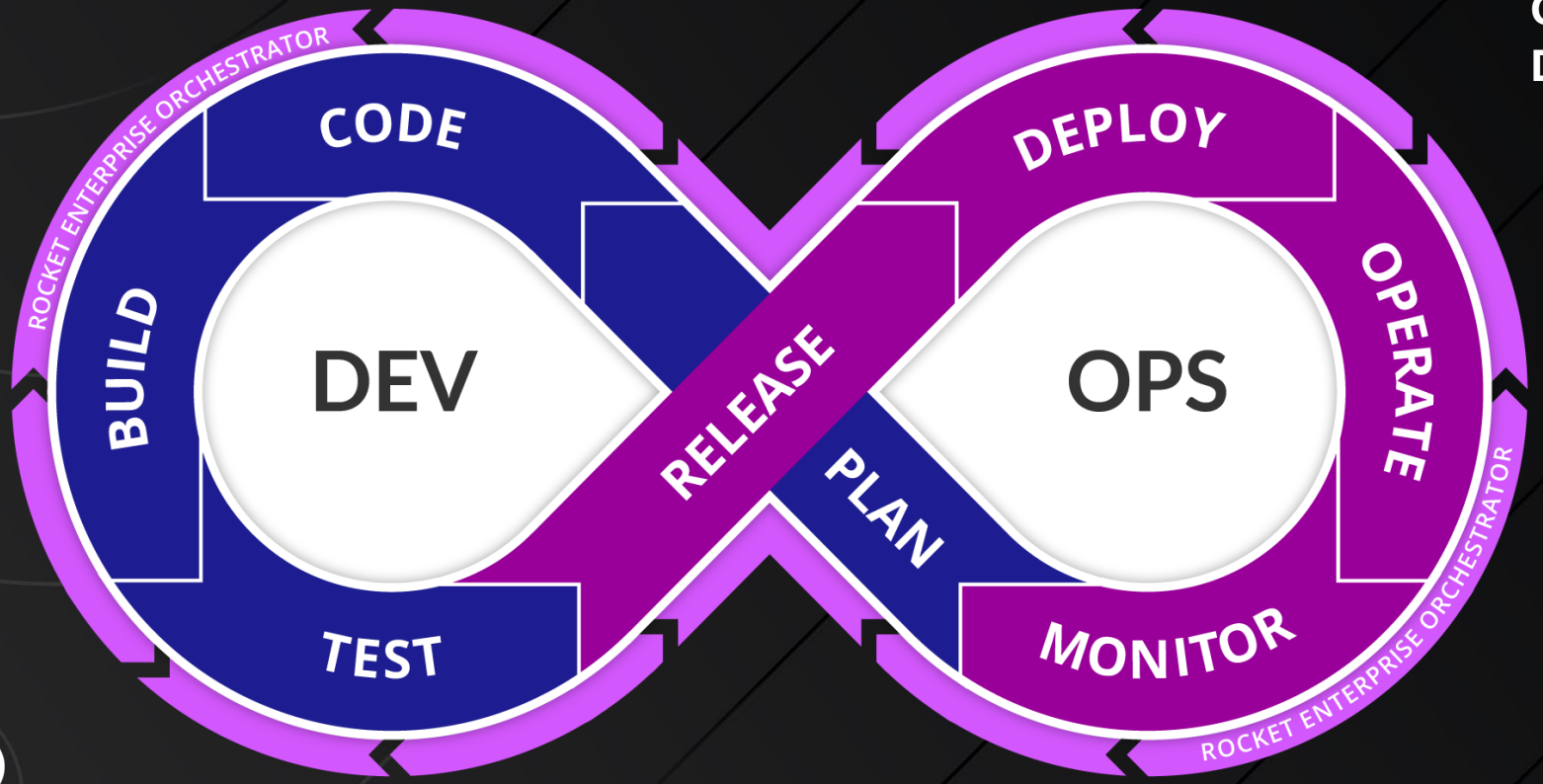
Automation is key

Iterative App Modernization requires DevOps





IBM i+ DevOps



Continuous Deployment (CD)

Continuous Integration (CI)

Benefits of clear CI/CD strategy

- Faster time to market and feedback from users
  through automation of the test process

- Improve code quality and reliability
  through more thorough testing during development
  by making testing cheaper and easier

- Empowering teams with valuable test result data and reporting
  so they can make informed business decisions

API Management Features

- **API Policy Manager**
- **API Analytics**
- **API Gateway**
- **API portal**
- **Deployment Flexibility**

Takeaways

- **Strangler Pattern** is a good strategy to break up the monolith
  Modernize incrementally and avoid the pitfalls of major rewrites
- **Test** thoroughly and often
  Shift left testing, required for CI/CD, will increase software quality & reduce costs
- Implement comprehensive **automation** technologies
  Automation is the key and will reduce time-to-production, errors, & skills reliance
- Ensure you achieve **transparency** throughout all development activities
  Visibility has never been more important with ever increasing complexity
- A good **DevOps strategy** will help you achieve this
  It's no longer optional

Q&A

Rocket® software

Disrupt.
Without disruption

No matter where you are in your modernization journey, Rocket Software has the right expertise and solutions to get you to where you want to be.

Swing by **Rocketsoftware.com** to see how we can help you **disrupt**, without disruption

Booth Number #515

COMMON
LEARN. ENGAGE. GROW.

POWER UP 2025
ANAHEIM, CALIFORNIA
MAY 19-22

Thank you

rocketsoftware.com
info@rocketsoftware.com

Rocket® software

Visit us in
Booth
#515