Fineract for Enterprise customer

The story of making a small open source project meet the need of a giant
The pitch

Fineract isn't for enterprise customers... It's for enterprise. It is used by large enterprises today. Although customers aren't the enterprise itself, don't get disheartened, it's going to be a good story. A story of the success of a small open source project.
Fineract is an open source core banking system, (also known as system of record [SOR]) with open API. It supports any banking product, service, or lending methodology. It supports accounting, general ledger, saving, loan or checking account management.
The customer

US based multinational financial technology company

More than

30K

employees

over

$25B

revenue

Part of top

100

Nasdaq
Why Open source?

✅ No vendor lock
✅ No license fee
✅ Control over destiny
❓ Performance
Performance

API performance
500 TPS with API response < 1000 ms

Batch processing
Close of business
4M loan accounts within 1 hour
PoC of Performance

Both the API and the batch processing were assessed. The actual db instance it was running on was db.r6g.16xlarge (ARM, 64vcpu, 512g ram) and db.r5.16xlarge (Intel, 64vcpu, 512g ram)

Outcome:
Sufficient API performance and double batch processing time. Both were in addressable range
So what is this story about

Performance, Scalability and Technology;
Quality assurance;
Team;
Business understanding;
Governance;
Tech stack upgrade

- **Postgres** to enable performance, partitioning,
  - Better performance
  - Possibility of partitioning
  - Keep supporting MySQL, MariaDB

- **Persistence layer OpenJPL to EclipseLink**
  - OpenJPL reached its end of life
  - Reference implementation following GPA
  - Better performance
Scalable and robust technology

- **Spring Batch** for job management
  - Parallel processing of multiple chunks to close loans as soon as possible
  - Prevention of conflict with online transactions
  - Steps to execute must be configurable
  - Retry and error handling
  - Catching up

- **Persistent event framework with Kafka or AMQ**
  - Replacing earlier JMS implementations
  - Support diverse needs
  - Event based data sharing with downstream systems
  - Replacing ETL jobs
Control your destiny - ability to attribute events to time defined by yourself

Close of Business - physical and logical date differentiation

Data recovery - well defined point of recovery
Quality

Problem with open source... it’s the source. Not the whole process of creating business value

On top of the existing integration tests we introduced a business case based regression test framework using gherkin. It allowed us to customize and differentiate smoke tests, regression tests.
Team

Community is key to find the right people. This is a network of individuals with diverse skills and it isn't limited to Fineract.

01 | Java developers
02 | Fineract knowledge
03 | Financial market understanding
04 | Technical leadership skills
Business understanding

One of our challenges is understanding the business needs.

Specific to US market (payments and transfers, legal requirements etc)
Each change requires a design decision what serves the open source community and what’s proprietary function.
Governance

Small agile team meeting big company standard timeline
The larger the company is the more dependencies it has to manage. These dependencies can explode your schedule.

Planning
You can plan as much as you understand.

Managing timeline and expectations
It's not only about highlighting issues but sharing those early. Management doesn't have a problem learning issues. They have problem learning them late.
Functional requirements

cob  event handling  idempotency

delinquency tagging loans

cob  event handling  availability

spring batch

event handling  delinquency tagging loans

external id  additional business event

write separation

business date  effective date

multiple disbursements  spring batch  eclipseLink  date

delinquency  idempotency
Let’s see where we ended up...
Results

1. Project delivered on time
2. Quickest production ramp up ever (in 4 weeks from 0 to 100%)
3. Without any notable issue
4. Over $200M projected cost saving over 5 years
Takeaways

1. Prove your application
   It proves you as well...

2. Harness the network of the community
   It’s a great network. Go even beyond. The more diverse

3. Adjust to expectations
   Not only technical but how you manage the project will impact the outcome. Manage time and quality. Guard the OS to remain beneficial.
Thank You