Our Good old team

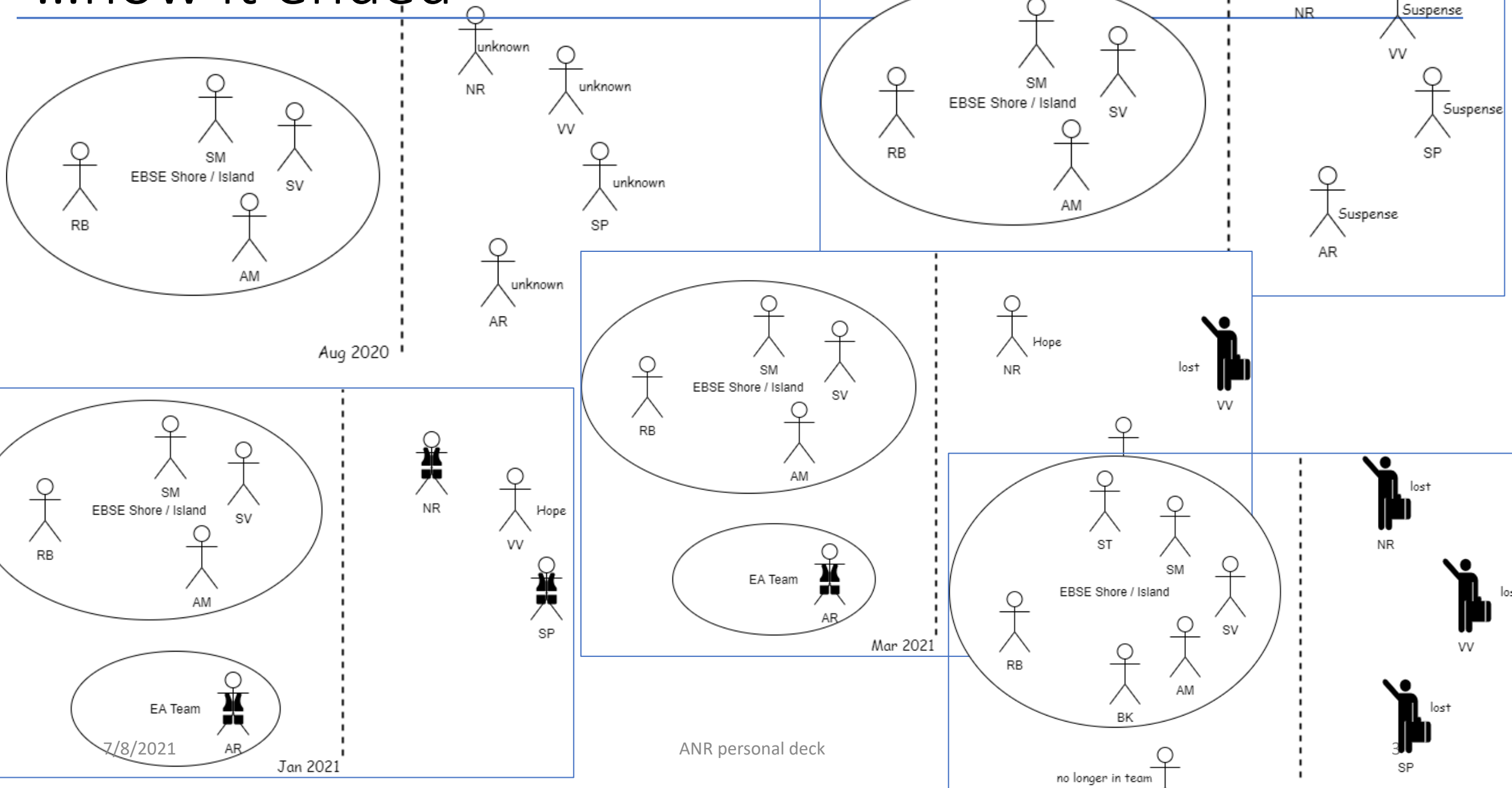
Learnings from the past

Last updated: July 8, 2021

Why this deck

- We had so many decks for our project: Intro deck, AOD for integrating teams, internal AOD for our team, Follow the data, Network components, Retrospective decks, etc.
- We had covered almost all aspects of application, team, and integrations.. Each area was thoroughly documented and kept up-to-date.
- But the circumstances beyond our control, fractured and broke our team in a bad way.
- Even though as a team as well as individuals, each one of us contributed above and beyond to produce an application and support system of highest quality standard, most of us were thrown out of the organization altogether – there was no proper closure/ farewell to the individuals that departed away from the team
- And hence this one-more-deck to bid adieu and reflect on the good memories/ learnings that we can take with us

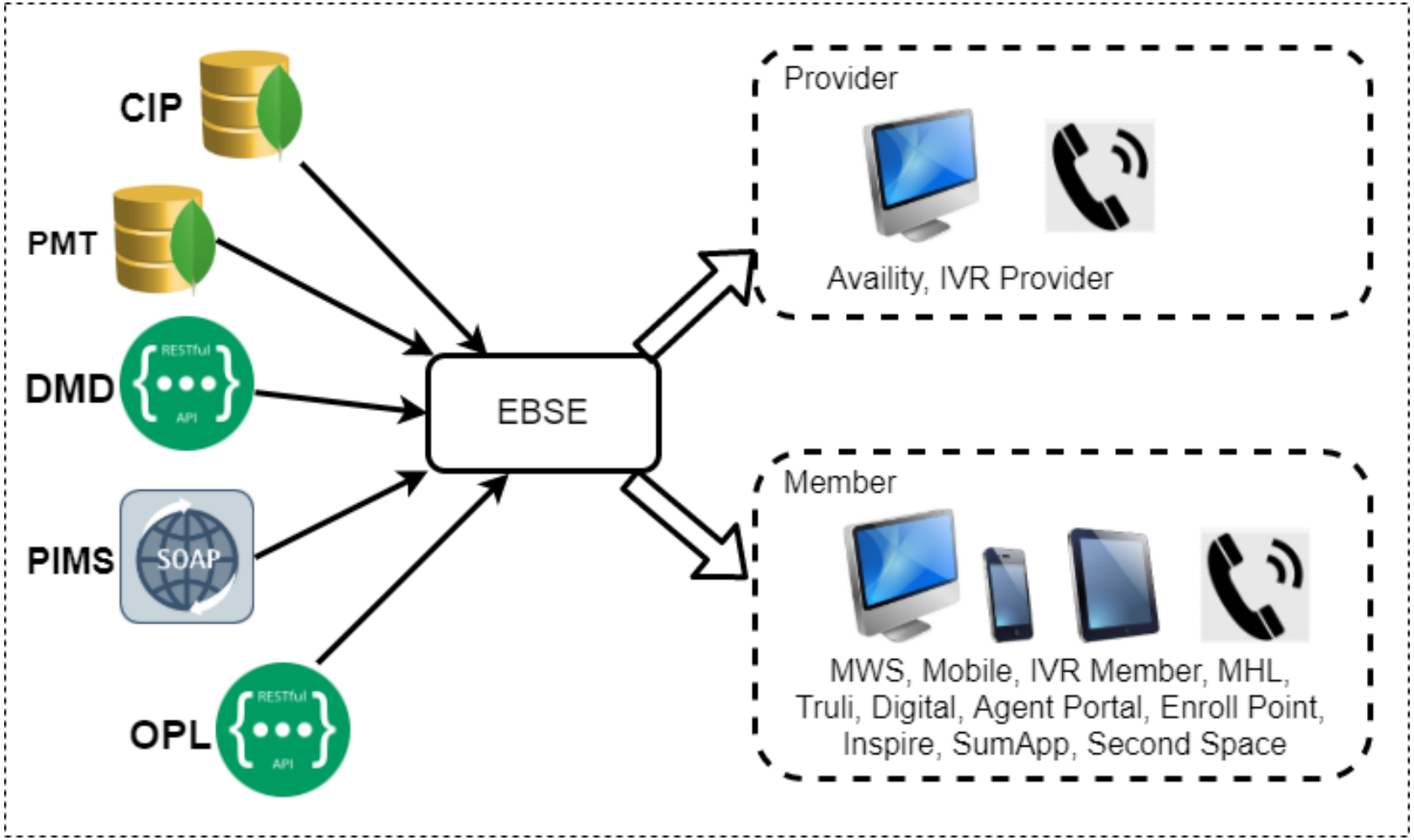
…how it ended


unknown
NR
unknown
VV
unknown
SP
SM
EBSE Shore / Island
SV
RB
AM
unknown
AR
Aug 2020
SM
EBSE Shore / Island
SV
RB
AM
Suspense
NR
Suspense
VV
Suspense
SP
Suspense
AR
Hope
NR
lost
VV
SM
EBSE Shore / Island
SV
RB
AM
EA Team
AR
Mar 2021
NR
Hope
VV
SP
SM
EBSE Shore / Island
SV
RB
AM
EA Team
AR
Jan 2021
ST
SM
EBSE Shore / Island
SV
RB
AM
BK
no longer in team
lost
NR
lost
VV
lost
SP

greenfield project

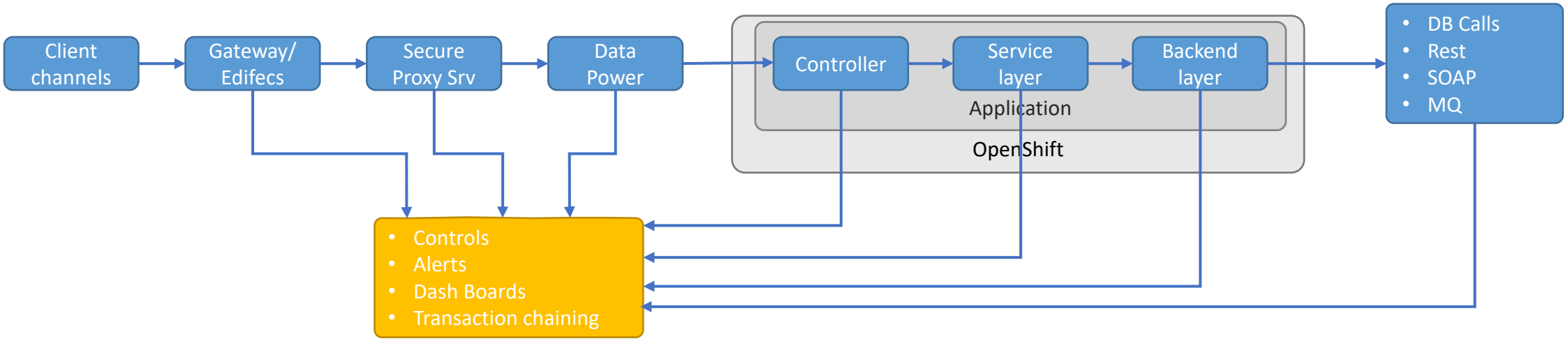
- The legacy E&B was served by Mainframe system for more than 30 years; several attempts at replacing that with modern API had failed
- EBSE was started to bring down mainframe maintenance cost, serve benefits from source systems, to be scalable and with top class control/ alerts
- It was established by a new team formed by borrowing from PMT and hiring new members
- On application design this followed many firsts within the organization, which included: SpringBoot, Gradle, MongoDB, OpenShift, Jenkins to state a few; along with parallel calls, circuit breakers, auto-scaling, rolling deployments, etc

a simple design



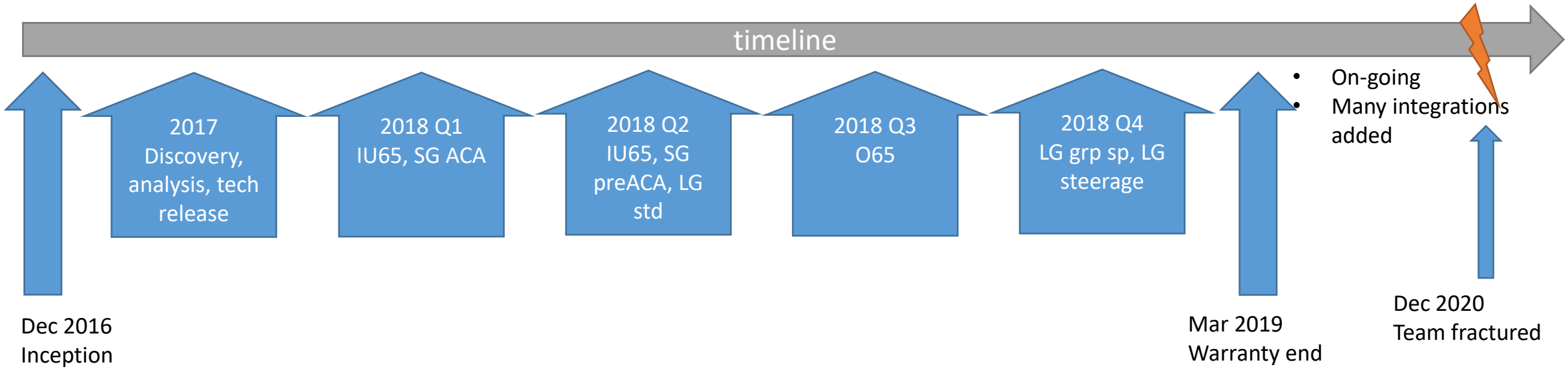
- Components were segregated by broad channel types
- The API service for tier 1 channel was kept separate from others
- Even though there were total of 4 API services, they were kept as a single monolithic application
- It was not un-necessarily broken into micro services architecture

attention to details



- Key aspect of the application design was intercepting at each critical point of call invocation and logging essential data
- Call chaining from invocation, starting with client app all the way through each hop was so important in giving valuable insight

phased approach

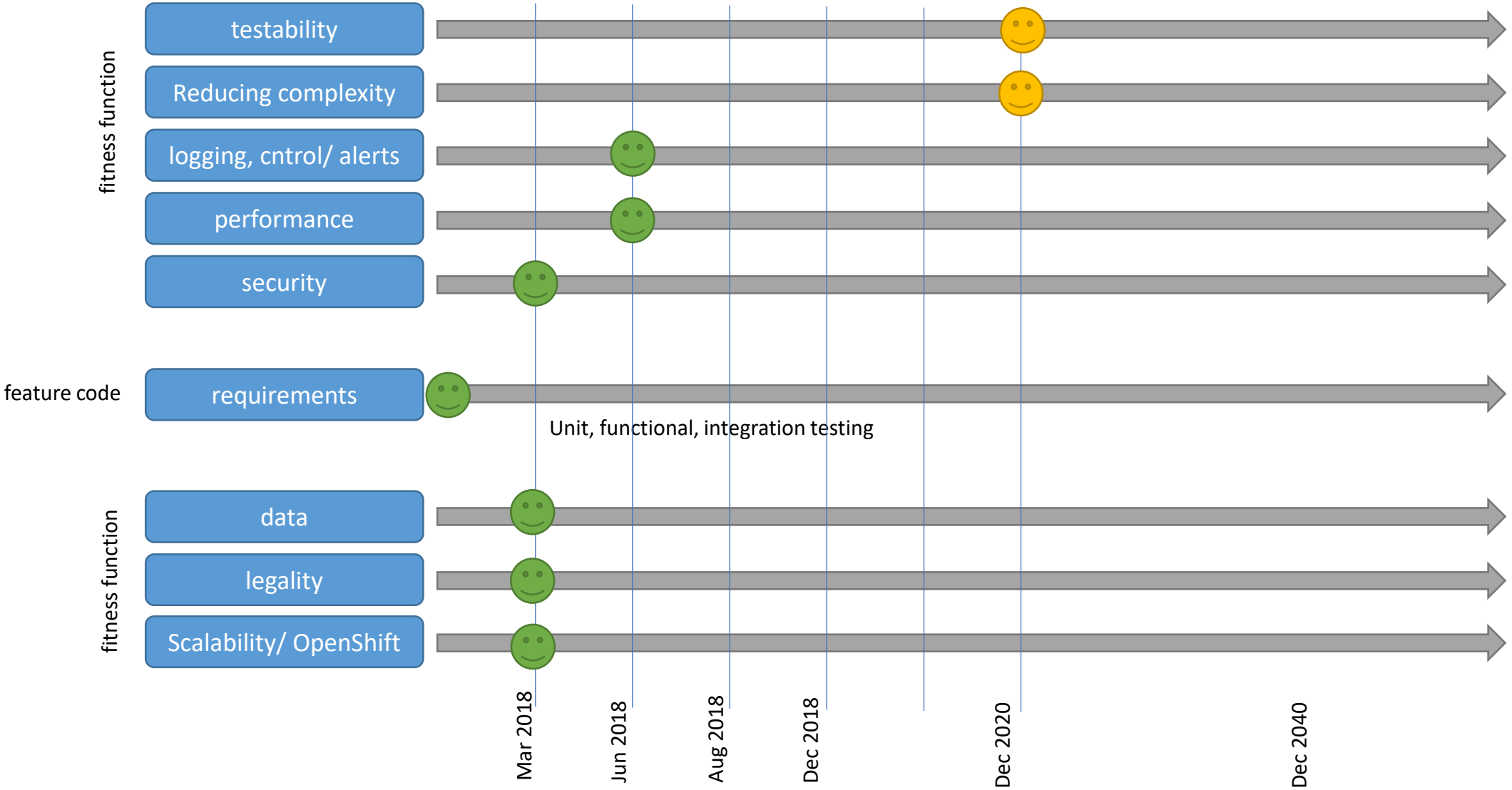


- Among the key aspects of implementation was a large discovery work; analyzing each aspect of mainframe application
- Technical implementation of API was completed first followed by thorough testing of build infrastructure on OpenShift
- Delivery was divided into phases: the new App was progressively loaded with transactions per segment per phase, keeping Mainframe app serving the rest traffic in parallel
- After all phases were transitioned to new App, and warranty period was crossed, the mainframe app was retired
- Later, the new App onboarded several new integrations, expanding the clientele of the services provided

operational excellence

- Maintaining a bin list of improvement items
- Taking up these improvements at slightest opportunity
- Continuously monitoring production system and identifying improvements
- Regularly scanning code through SonarQube and refactoring the application
- Maintaining up to date documentation of the design as well as the process around the application support
- Interacting with interfacing teams .. Being responsive to queries and tickets and establishing a solid rapport with all

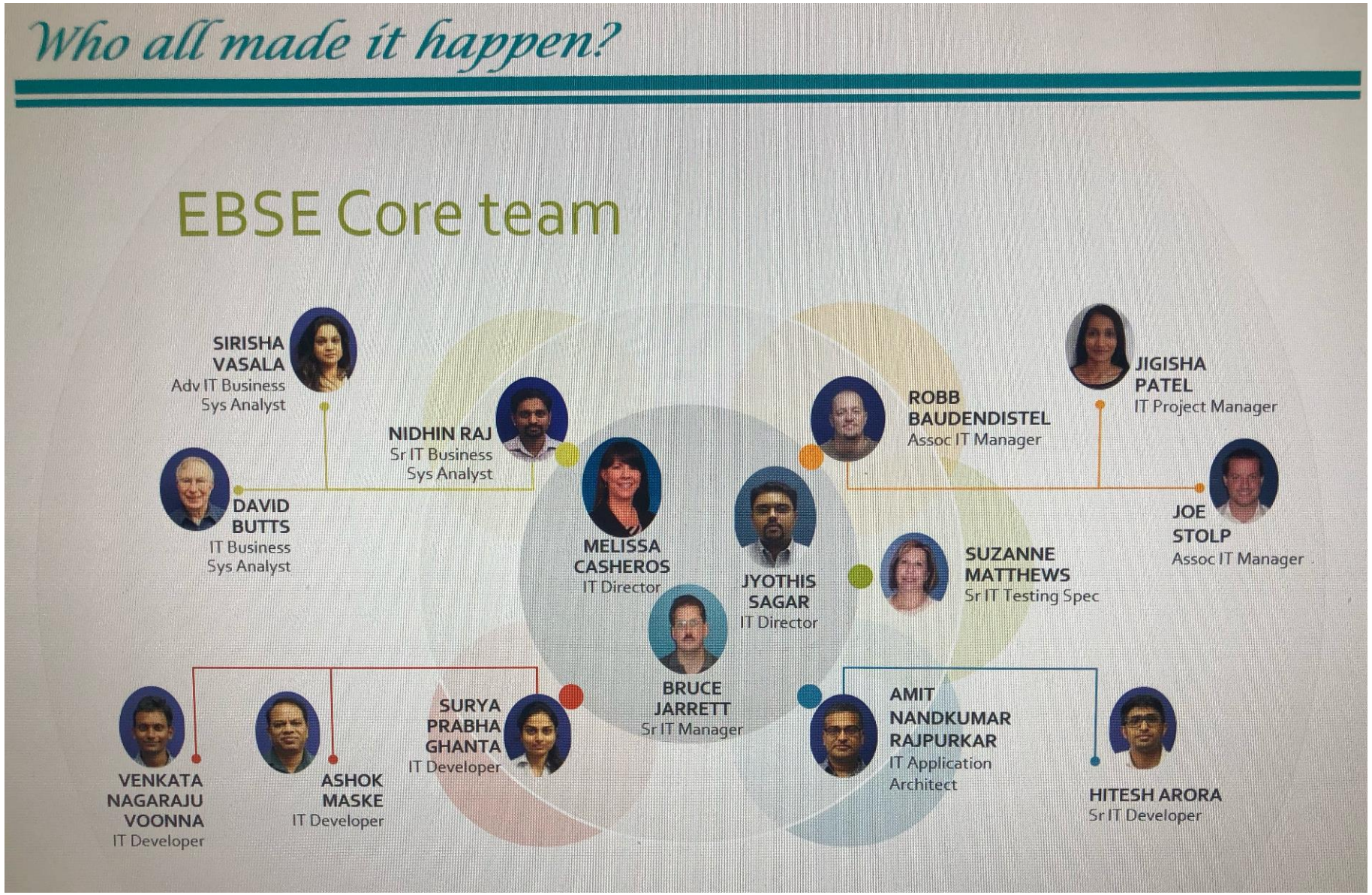
Shelf-life vs non-functional code



The Magic team

- Dedicated Architect working at highest level of abstraction
  - Constantly recording documentation and always keeping it up to date
  - Providing direction, theme/ central idea to the team
  - Main theme = 24x7 availability and decades long shelf life; “operation support is at the heart of the application design”
- Analysts working at deep dive level; thorough understanding of source system data and data-wiring
  - Benefit translation and complete understanding of source data
  - How the translated data is consumed by downstream/ its context
  - Exceptional ops support, data-analysis with depth of knowledge
- QA ensuring quality of information served at Availity as well as matching with Expectation from Business
  - The only limitation was using manual testing from external UI App; remaining testing was driven by developers
- Developers working at lowest level of abstraction
  - Each developer pays close attention to quality of service; developers not only taking deep dive within the realm, they also went out of realm
  - Team is providing SME-expertise to upstream and downstream channels

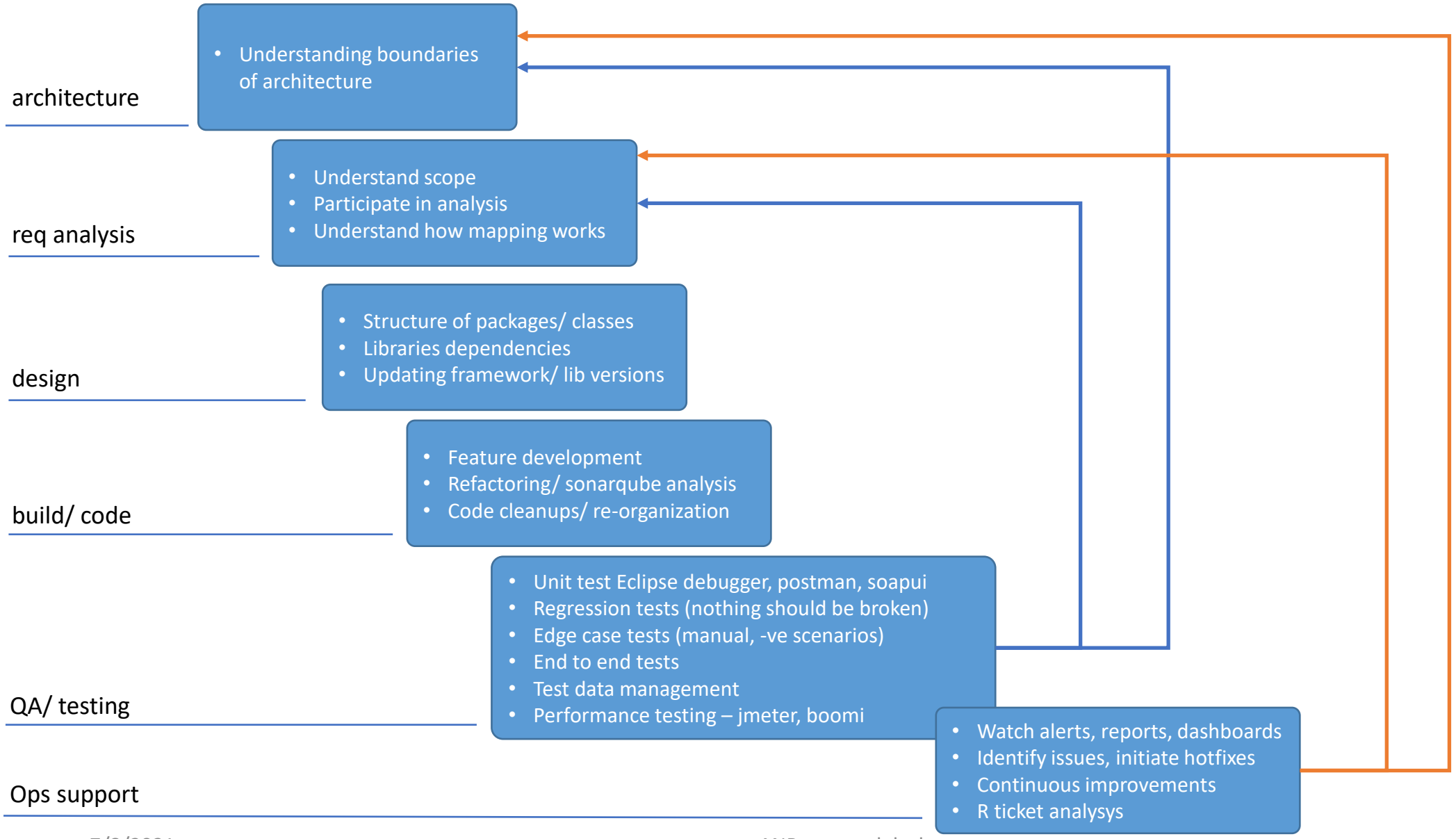
The team


Who all made it happen?
EBSE Core team
SIRISHA VASALA
Adv IT Business Sys Analyst
NIDHIN RAJ
Sr IT Business Sys Analyst
DAVID BUTTS
IT Business Sys Analyst
MELISSA CASHEROS
IT Director
ROBB BAUDENDISTEL
Assoc IT Manager
JIGISHA PATEL
IT Project Manager
JOE STOLP
Assoc IT Manager
JYOTHIS SAGAR
IT Director
SUZANNE MATTHEWS
Sr IT Testing Spec
BRUCE JARRETT
Sr IT Manager
VENKATA NAGARAJU VOONNA
IT Developer
ASHOK MASKE
IT Developer
SURYA PRABHA GHANTA
IT Developer
AMIT NANDKUMAR RAJPURKAR
IT Application Architect
HITESH ARORA
Sr IT Developer

The developers

- Complete handle on code base and self driven
  - Not just expertise on java, spring framework and related technologies
  - Having insight and adhere to clean code practices, actively working on simplifying/ improving code; use SonarQube
  - Developers distribute workload amongst themselves and deliver features end-to-end (code-complete)
- Full understanding of data, how/ why it is used as per design
  - How benefits are designed in source system and how data is made available in cache db
  - How the member data is designed and made available in respective cache
  - How the accumulator data was arranged with benefits
- DevOps oriented
  - Each developer was an expert working with Gradle, GIT, Jenkins,
  - Work along side infra teams, gain expertise on OpenShift
- Always going above/ beyond in testing/ QA
  - Fill up void of test team; not just unit test but also end-to-end functionality test
  - Test from perspective of end channels too
- Test data arrangement
  - Work along side organization teams and help arrange test data for the application too
- Full grip on underlying source app and the tier 1 client app
  - Keep a thorough understanding on how the response data appears on the tier 1 client

Role played by developers



The team on the whole

- Self managed team
  - Project planning, task assignments, delivery, task-tracking was all managed by team members themselves
- Working shoulder to shoulder
  - Developers distribute their workload
  - Developers work hand in hand with analysts and QA
- Informal code reviews/ developer sessions
  - Being proactive in reviewing with each other any key deviations in code/app-design
- Design sessions
  - Analysts will actively have design sessions with developers for any new requirements
  - Many times developers also drive design sessions, as feedback loop after digging through implementation
- Continuous improvements
  - Each team member contributed to continuous improvement
- Contributing above/ beyond
  - None of team members settled with 9-5 job,
  - Each one contributed into areas beyond their individual scope to make End to End possible
  - Many worked after hours too for additional research/ analysis to achieve improvements
- Delivering Quality of Service as a whole
  - The team overall, checked on each aspect of Quality of Service

Quality of Service

Quality of Work

- Value of Work
- Delivered by individuals, teams
- This can include the quality of task completion, interactions, and deliverables

- Fit for purpose
- Conformance to requirements
- Completeness
- Correctness
- Accurate
- Diligence
- Professional
- Communication
- Compliance
- Controls
- Best Practices
- Risk
- Integration
- Usability
- Customer friendly
- Relevant

- https://simplicable.com/new/work-quality

What could have been better

- The invaluable talent that was available as top-quality self driven team members should have been kept
- Dedicated resources which were lost are not replaceable
- Even the immediate supervisors were kept in dark about the team members tenure
- The uncertainty and lack of information/ foresight was the bad part

- Had it been known that organization was only going to throw away valuable team members, we could have ended in better way

On a positive note..

- The experience that each team member gathered over these years on this project has been invaluable
- The success stories (several of them) that each one experienced will remain as good guidelines for us for future reference

- The team spirit seen by this project team is something that each one will try to re-create in new environment