Modernizing Legacy Software

Jared King
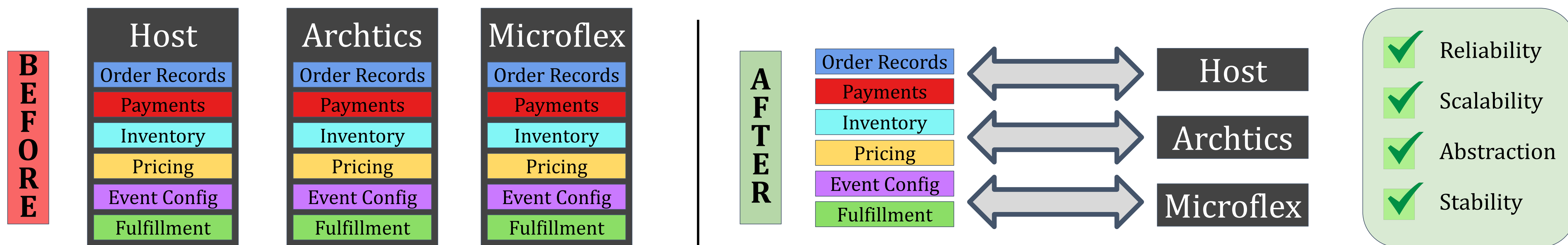
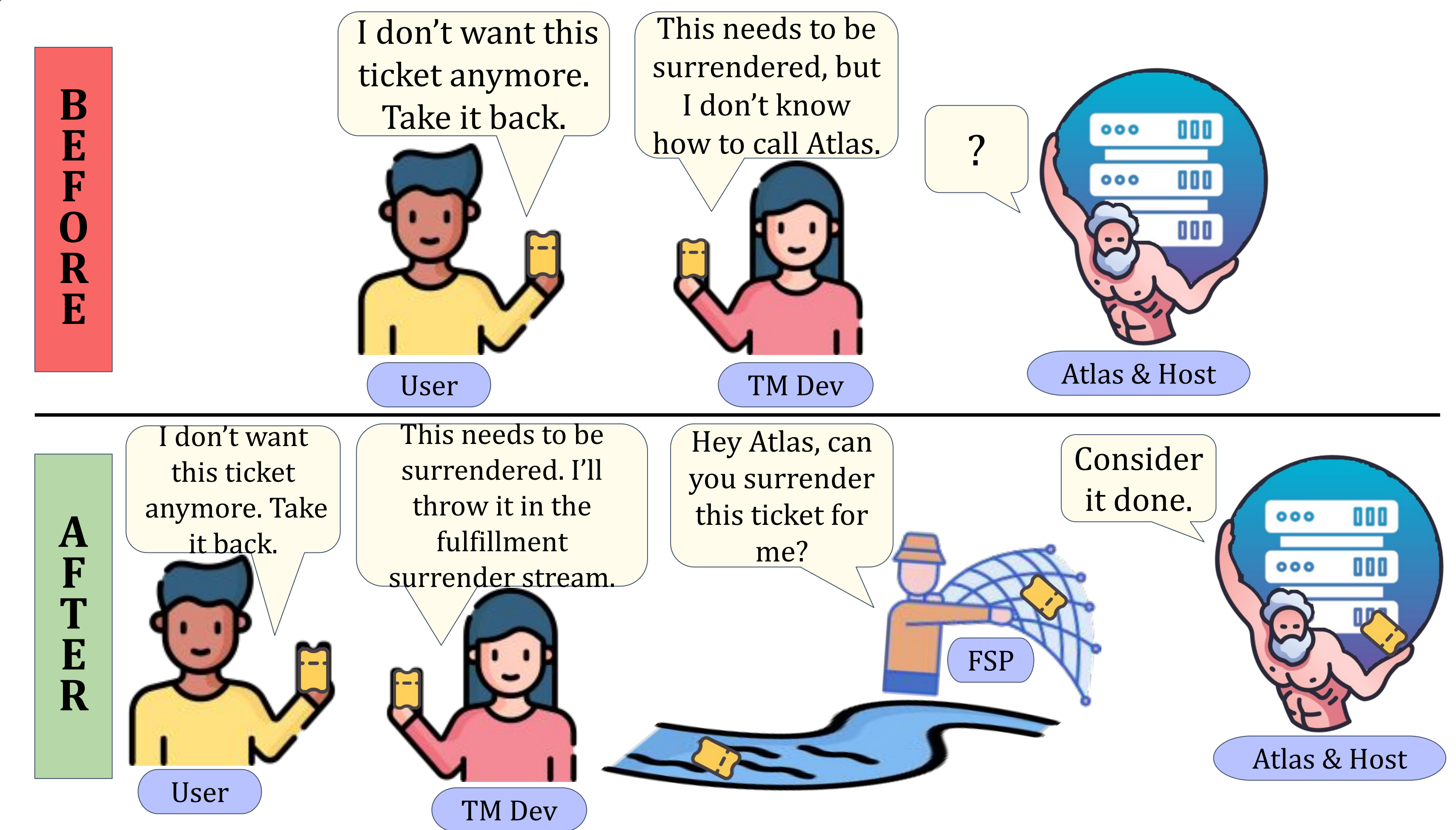*Computer Science, University of New Hampshire, Durham, NH 03824*

Introduction

Ticketmaster has multiple ticketing systems with unique, complex, tech stacks. One of these ticketing systems, The Host, is a legacy technology foundational to Ticketmaster's ticket sales. It faces the challenge of managing changes in seat ownership, particularly the burden of "surrenders." A surrender is when a ticket changes possession. Fulfillment Surrender Processor (FSP) is a dedicated stream processor that will move ownership management off the Host and onto a new, modern system.
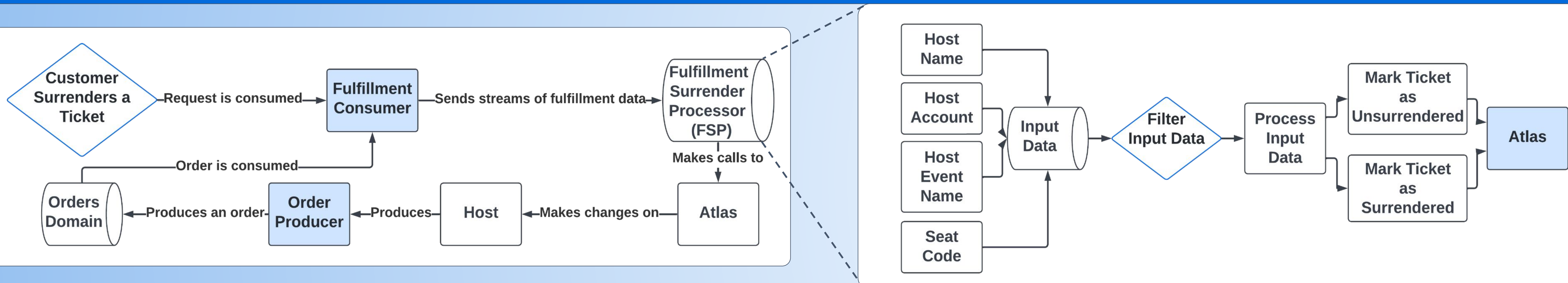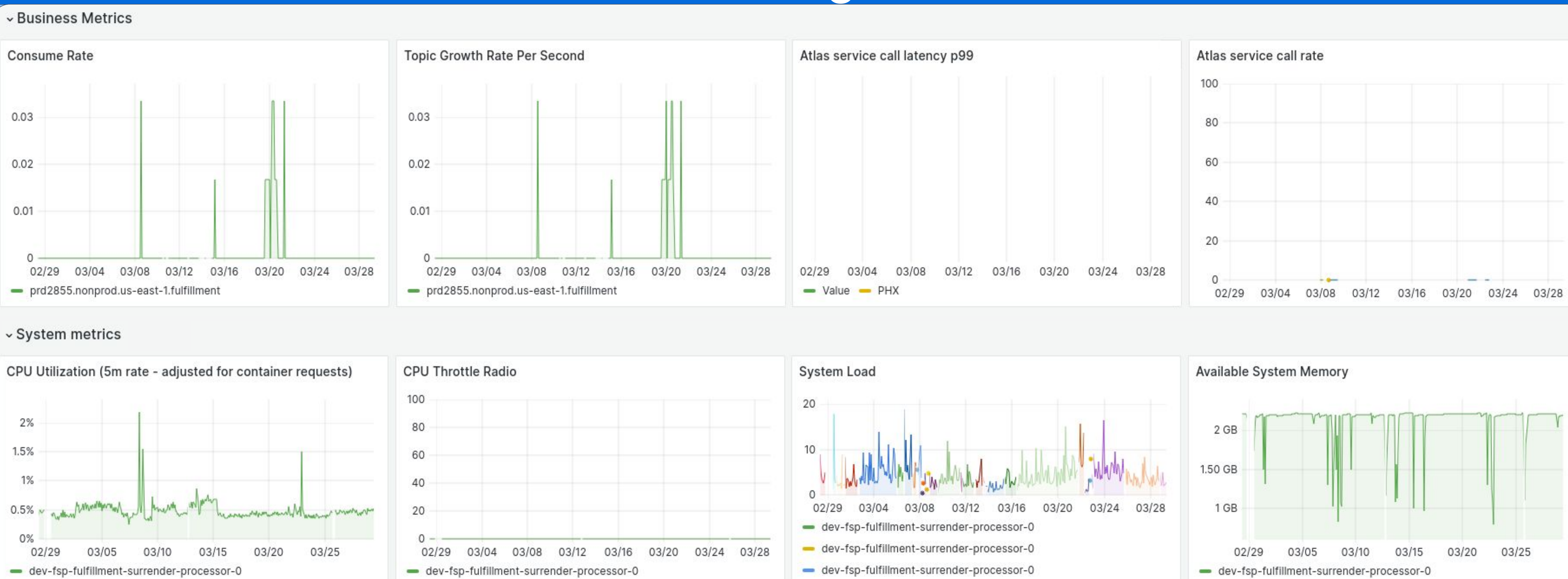
Why Domain-Driven Development?

BEFORE

Host
- Order Records
- Payments
- Inventory
- Pricing
- Event Config
- Fulfillment

Archtics
- Order Records
- Payments
- Inventory
- Pricing
- Event Config
- Fulfillment

Microflex
- Order Records
- Payments
- Inventory
- Pricing
- Event Config
- Fulfillment

AFTER
- Order Records
- Payments
- Inventory
- Pricing
- Event Config
- Fulfillment

↔ Host
↔ Archtics
↔ Microflex

✓ Reliability
✓ Scalability
✓ Abstraction
✓ Stability

Why FSP?

BEFORE

- User: "I don't want this ticket anymore. Take it back."
- TM Dev: "This needs to be surrendered, but I don't know how to call Atlas."
- Atlas & Host: "?"

AFTER

- User: "I don't want this ticket anymore. Take it back."
- TM Dev: "This needs to be surrendered. I'll throw it in the fulfillment surrender stream."
- FSP: "Hey Atlas, can you surrender this ticket for me?"
- Atlas & Host: "Consider it done."

How Does FSP Work?

Customer Surrenders a Ticket →(Request is consumed)→ Fulfillment Consumer →(Sends streams of fulfillment data)→ Fulfillment Surrender Processor (FSP) →(Makes calls to)→ Atlas

Order is consumed → Fulfillment Consumer

Orders Domain →(Produces an order)→ Order Producer →(Produces)→ Host →(Makes changes on)→ Atlas

Host Name, Host Account, Host Event Name, Seat Code → Input Data → Filter Input Data → Process Input Data → Mark Ticket as Unsurrendered / Mark Ticket as Surrendered → Atlas

Monitoring FSP

▾ Business Metrics

Consume Rate
0.03 / 0.02 / 0.01 / 0
02/29 03/04 03/08 03/12 03/16 03/20 03/24 03/28
— prd2855.nonprod.us-east-1.fulfillment

Topic Growth Rate Per Second
0.03 / 0.02 / 0.01 / 0
02/29 03/04 03/08 03/12 03/16 03/20 03/24 03/28
— prd2855.nonprod.us-east-1.fulfillment

Atlas service call latency p99
02/29 03/04 03/08 03/12 03/16 03/20 03/24 03/28
— Value — PHX

Atlas service call rate
100 / 80 / 60 / 40 / 20 / 0
02/29 03/04 03/08 03/12 03/16 03/20 03/24 03/28

▾ System metrics

CPU Utilization (5m rate - adjusted for container requests)
2% / 1.5% / 1% / 0.5% / 0%
02/29 03/05 03/10 03/15 03/20 03/25
— dev-fsp-fulfillment-surrender-processor-0

CPU Throttle Radio
100 / 80 / 60 / 40 / 20
02/29 03/04 03/08 03/12 03/16 03/20 03/24 03/28
— dev-fsp-fulfillment-surrender-processor-0

System Load
20 / 10 / 0
02/29 03/04 03/08 03/12 03/16 03/20 03/24 03/28
— dev-fsp-fulfillment-surrender-processor-0
— dev-fsp-fulfillment-surrender-processor-0
— dev-fsp-fulfillment-surrender-processor-0

Available System Memory
2 GB / 1.50 GB / 1 GB
02/29 03/05 03/10 03/15 03/20 03/25
— dev-fsp-fulfillment-surrender-processor-0

Results

Ticketmaster's overall tech stack has been simplified, leading to improved reliability and scalability. This will result in a streamlined development process.

Conclusions

Modernizing legacy software that has existed for this long and on a large scale is a complex task. Instead, modernizing must be done in increments when available time and resources allow it. The term "modern" is relative to the time, so the process of modernization must always be kept as a priority. Incorporating FSP not only aids integration with modern systems but also represents a pivotal step towards future-proofing Ticketmaster's infrastructure.

Acknowledgments