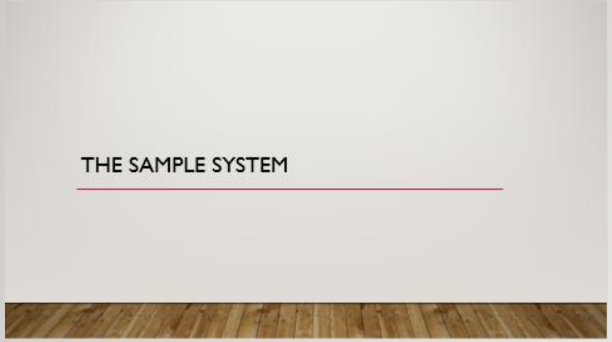
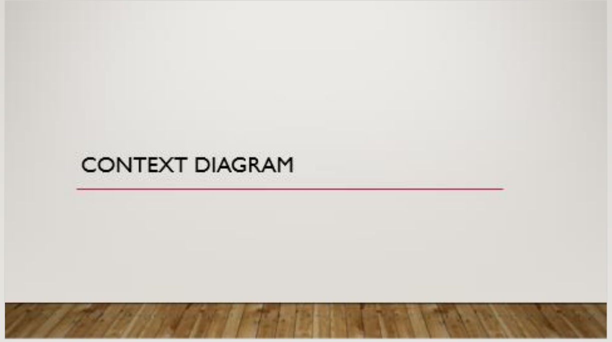
FUNDAMENTALS OF ARCHITECTURE



THE SAMPLE SYSTEM



CONTEXT DIAGRAM

ARROWS ACROSS THE BOUNDARY

ARCHITECTURE QUALITIES

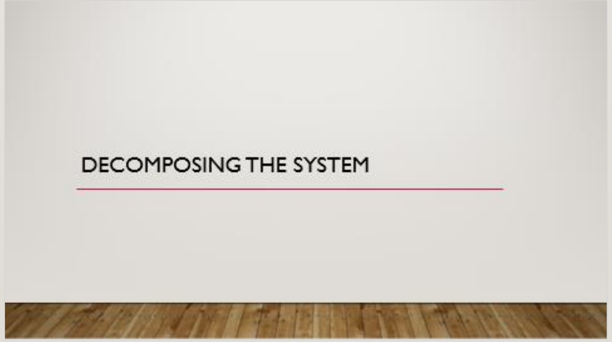
ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

ASRs

CONSTRAINTS

CHECKPOINT

Where are we now?



DECOMPOSING THE SYSTEM

VIEWS YOU CAN USE

C4 – Context, Containers, Components, Classes

THE SAMPLE SYSTEM

PURPOSE OF THE SAMPLE SYSTEM

- Allows me to illustrate with a real-ish system
- Allows us to compare solutions
- Relatively simple domain
- Enough complexity to be interesting

- Apologies in advance if this is your actual business plan

THE SYSTEM

- “Subscriptions as a Service” startup
- Supporting local businesses: dog walkers, dry cleaning
- “White label” sites
- We handle the money: bill customers, pay providers
- We handle customer service calls

PROVIDERS

- Set up their goods & services
- Define their service area
- Decide how often they can deliver (weekly, monthly, quarterly)
- But not inventory.We do not manage inventory.

SUBSCRIBERS

- Think they're dealing with the provider
- Choose a tier of service or package of goods
- Decide how often they want service
- Provide a payment method
- Decide whether to auto-renew

YOU

- Play the role of the technical co-founder
- It’s your money on the line
- Seed round investors are interested but wary
- They want to see you get to $1,000,000 ARR (annual recurring revenue)

ACTIVITY: WHERE DO WE START?

1. Pick a JavaScript framework
2. Decide the team structure
3. Shape the fundamental architecture
4. Explore the problem space
5. Make a decision matrix
6. Sign up for AWS and define your microservice APIs
7. Something else

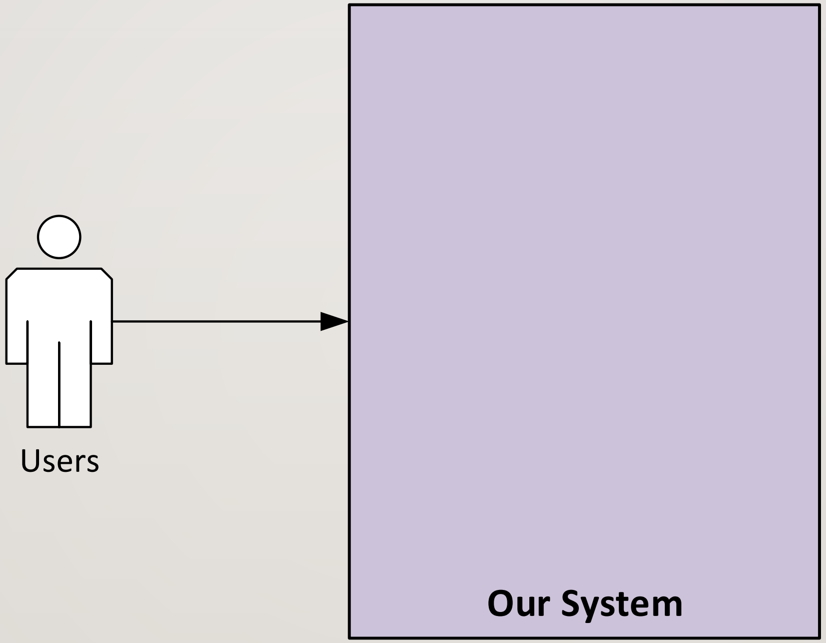
ACTIVITY: WHERE DO WE START?

1. Pick a JavaScript framework
2. Decide the team structure
3. Shape the fundamental architecture
4. Explore the problem space
5. Make a decision matrix
6. Sign up for AWS and define your microservice APIs
7. Something else

CONTEXT DIAGRAM

SYSTEM CONTEXT

- Every system exists in a context, defined by:
  - Users
  - Other systems


Users
Our System

LOOK FOR ALL THE CONSTITUENTS

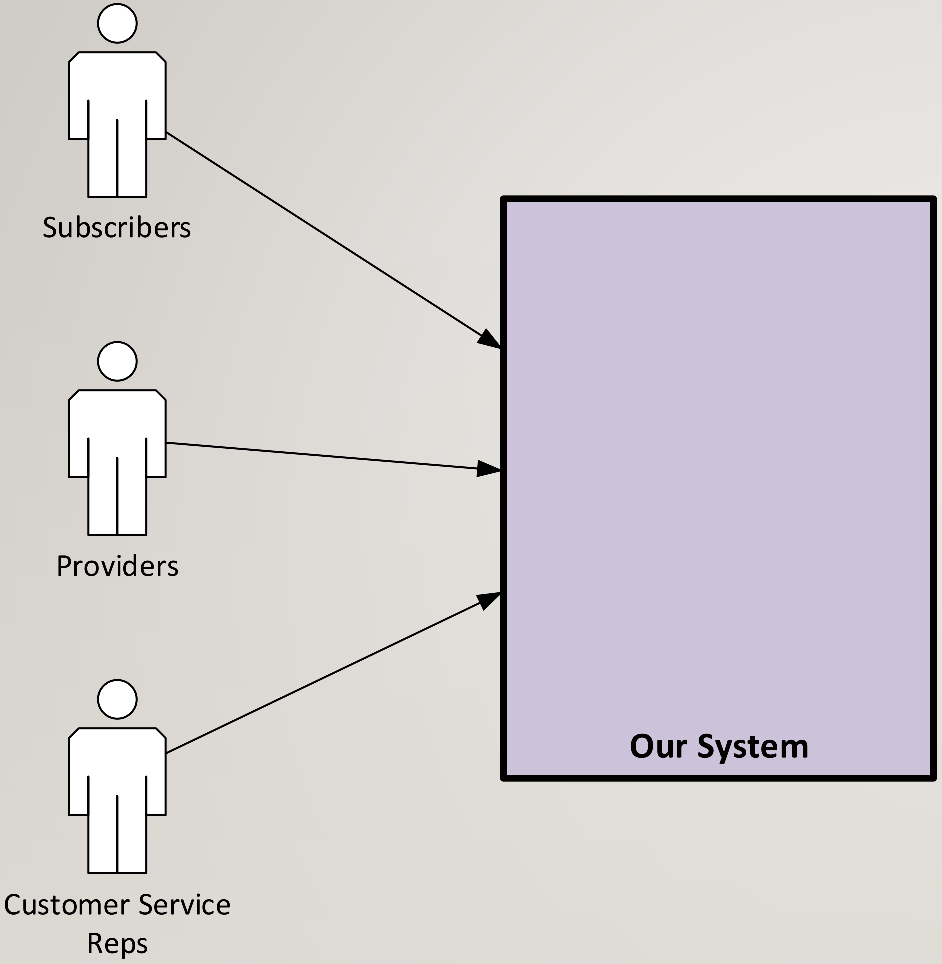
- Customers are easy to spot
- Look for users that facilitate business:
  - Customer service reps & managers
  - Finance staff
  - Sales
- Look for users that run the system itself:
  - Technicians
  - Operators & administrators

ACTIVITY: USER GROUPS FOR OUR SYSTEM

Name the user groups that you can identify.

Think about:

1. People who pay the bills
2. People who run the business
3. People who run the system



DECISION TIME!

How do providers get set up?

You choose:

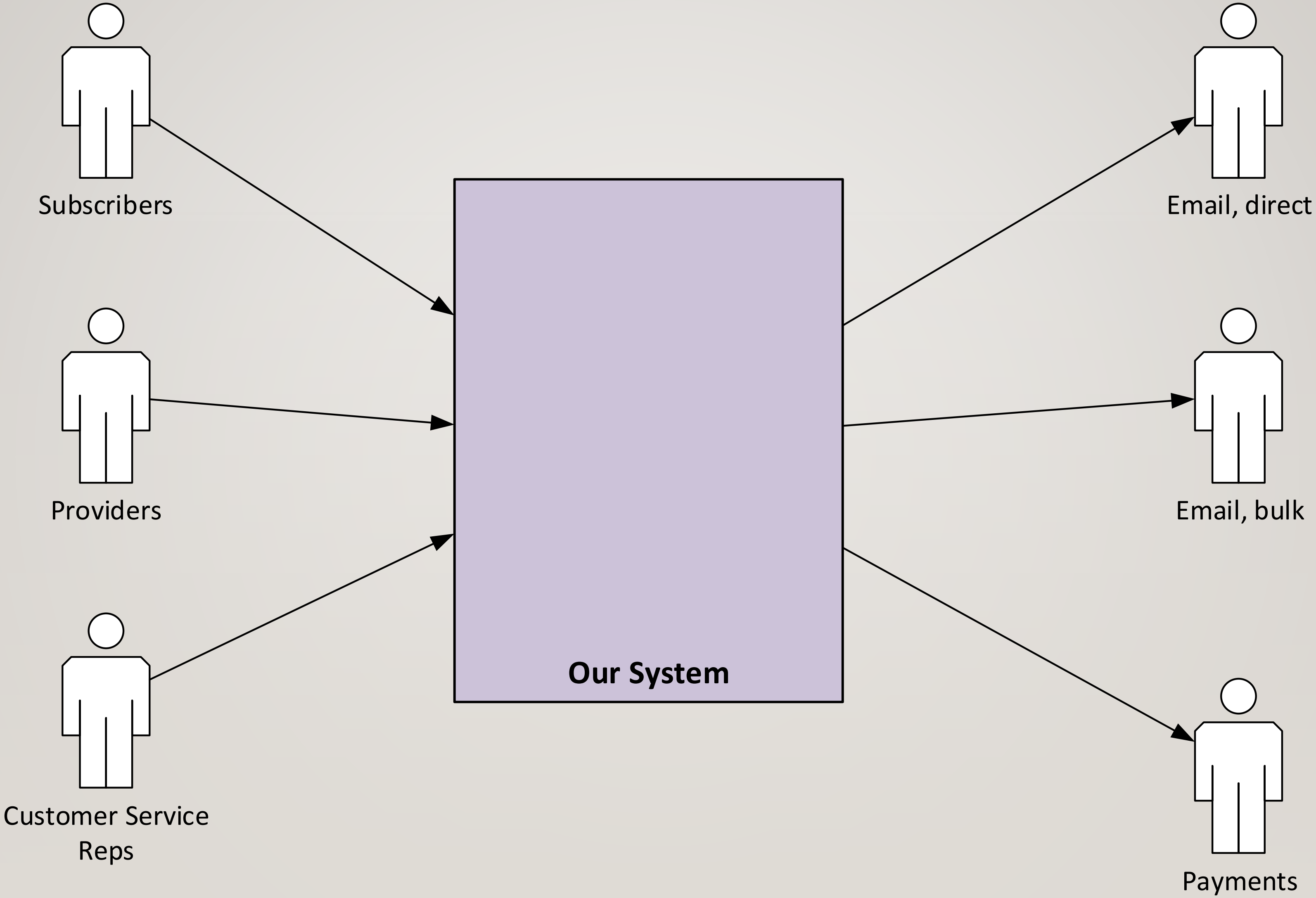
A. Self-service.

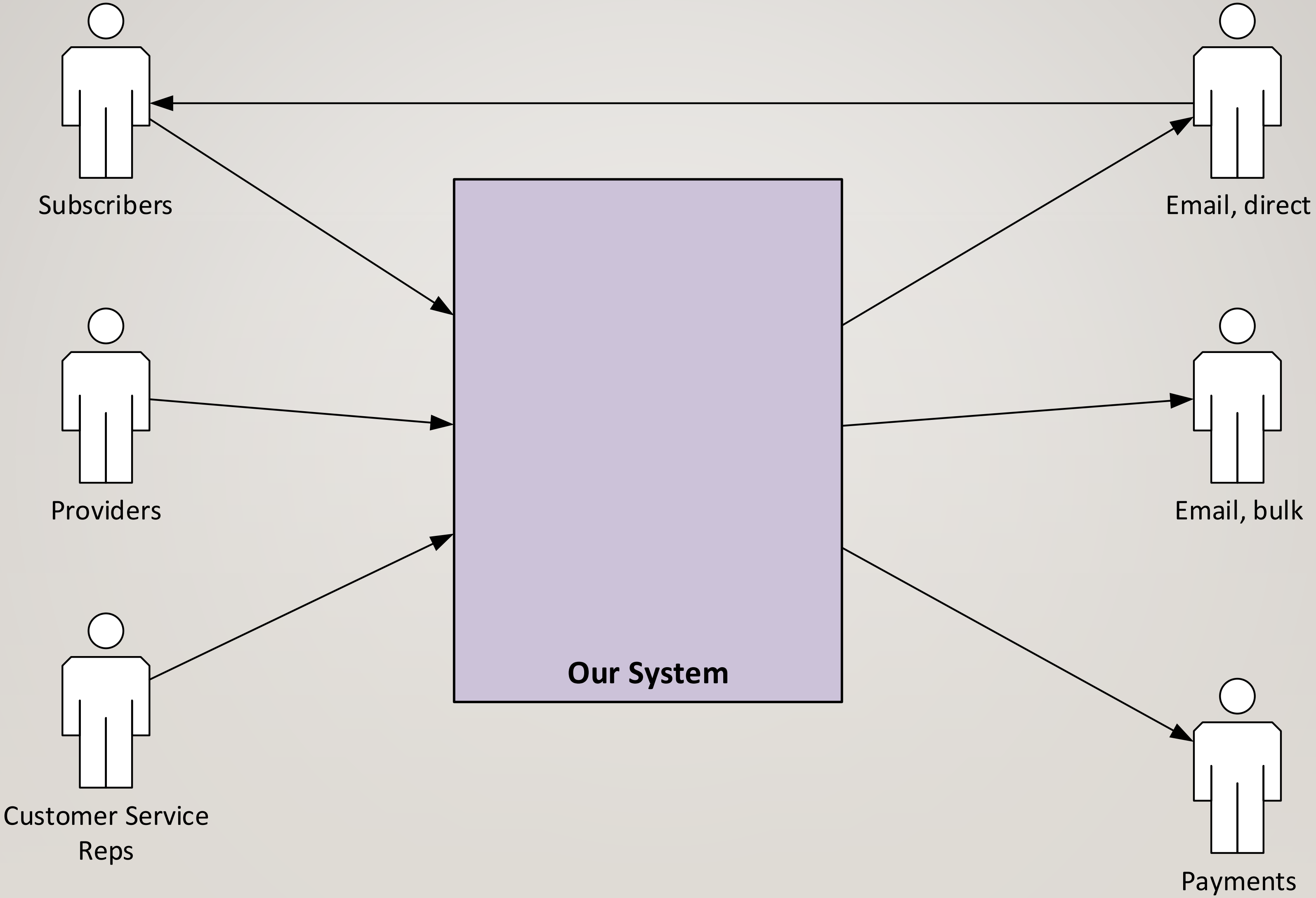
B. Sales reps on the ground.

If you chose B, then the sales reps are a new user group.

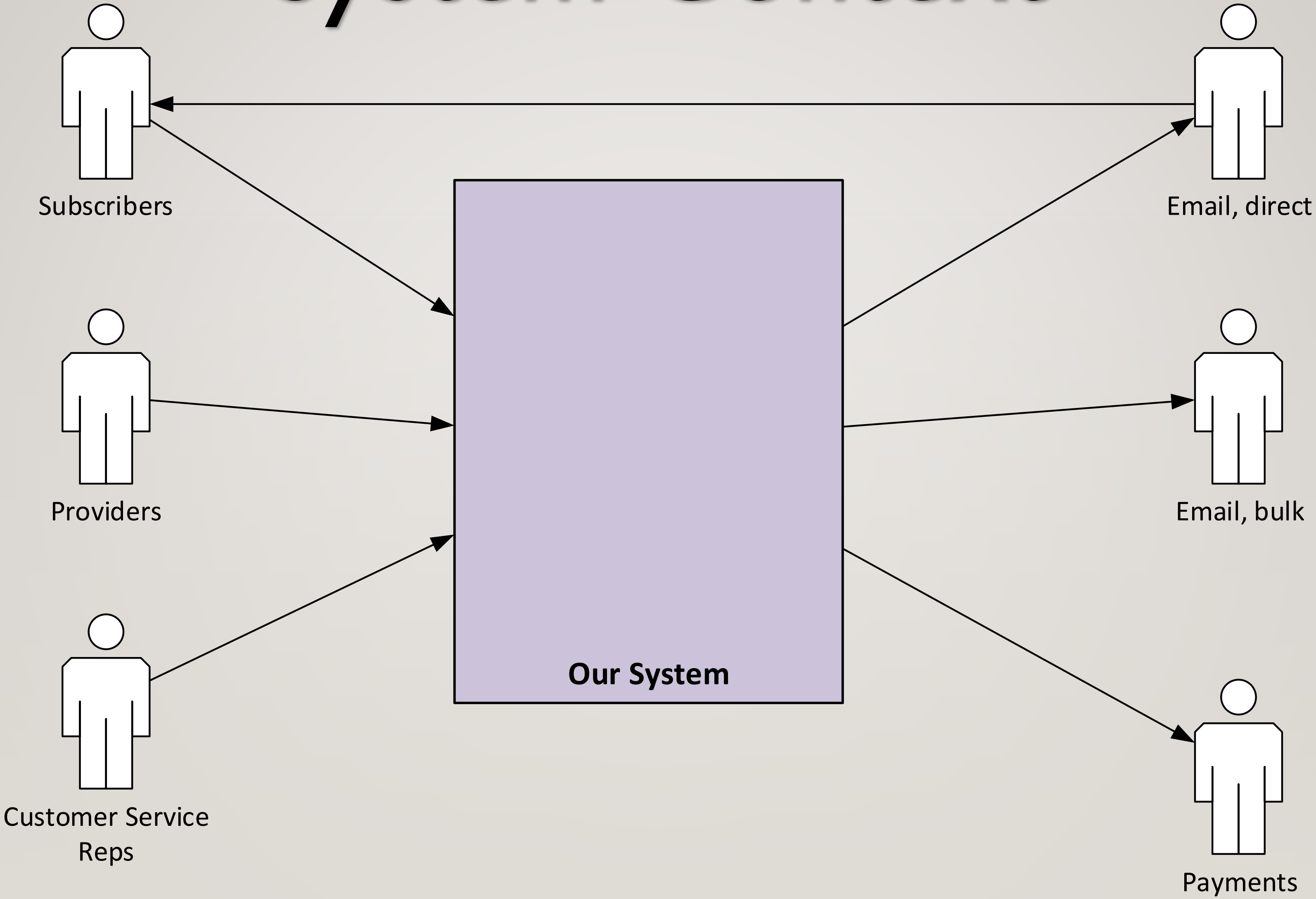
OTHER SYSTEMS: WHAT SHALL WE OUTSOURCE?

- Email, bulk and direct
- Payments
- Calendar?
- Currency exchange?
- Authentication?
- Authorization?
- SSO?
- Monitoring?
- Alerting?
- Fraud detection?


Subscribers
Providers
Customer Service Reps
Our System
Email, direct
Email, bulk
Payments


Subscribers
Providers
Customer Service Reps
Our System
Email, direct
Email, bulk
Payments

System Context



SYSTEM INTERIOR

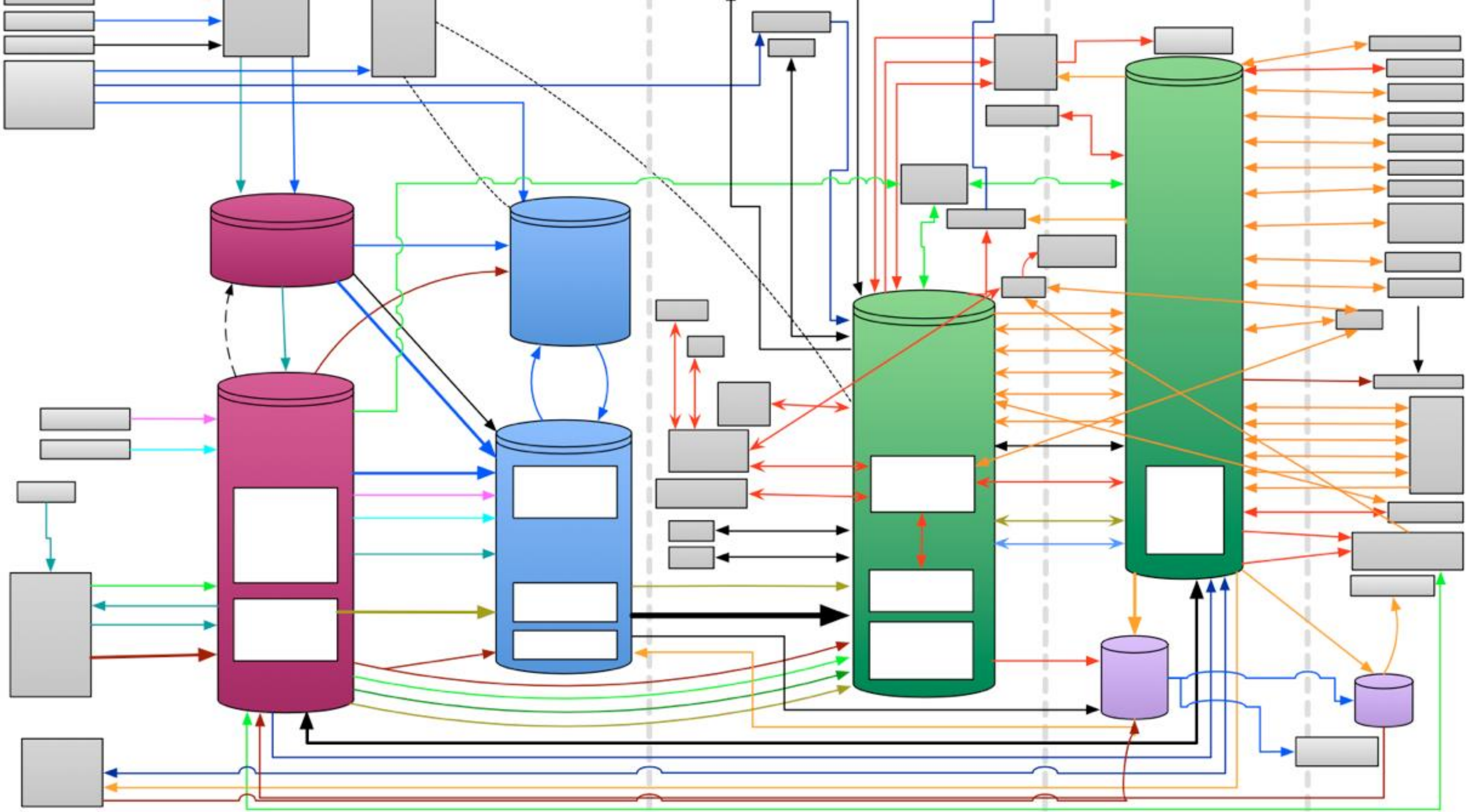
- Left blank on purpose
- For now, focus on the *interfaces*

ABOUT DIAGRAMS

- Diagrams, and documents in general, get a bad rap
- We do them to help discover and communicate, **not** because the process tell us to
- Keep them lightweight
- Provide a legend

VIEWS AND VIEWPOINTS

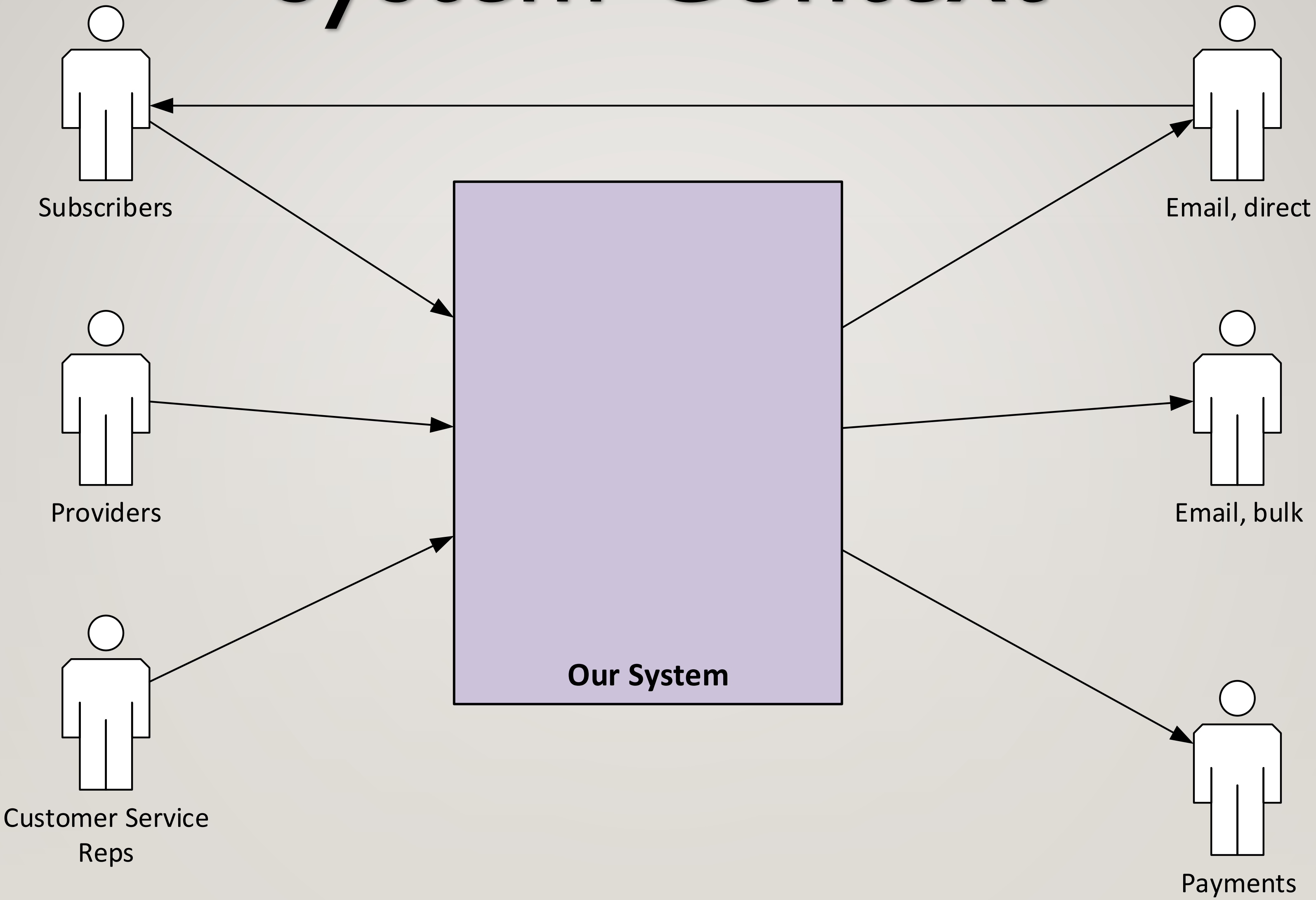
- Different team members have different needs
- One diagram **cannot** serve them all.



VIEWS AND VIEWPOINTS

- Different team members have different needs
- One diagram **cannot** serve them all.
- A *view* has a notation:
  - Elements
  - Relations
  - Constraints
- And a usage

System Context



SYSTEM CONTEXT VIEW

Elements	• User roles • External systems
Relations	• Uses
Constraint	• “Uses” has one endpoint on a user or external system

ACTIVITY: CONTEXT DIAGRAM

Create your context diagram

- Diagramming tools: Visio, OmniGraffle, Gliffy
- Text-to-diagram: PlantUML
- Whiteboard + phone camera
- Paper napkin

Have you found more user groups?

Are you integrating with more systems?

ARROWS ACROSS THE BOUNDARY

interface (n.)

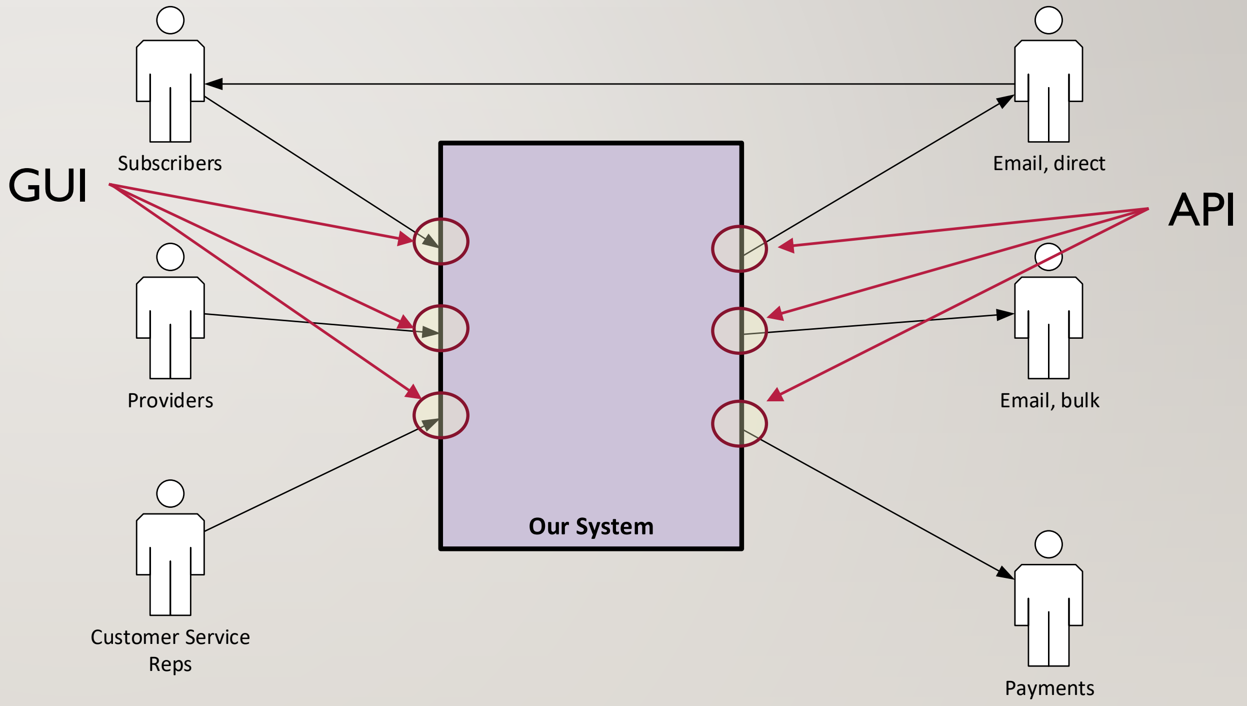
"a plane surface regarded as the common boundary of two bodies," 1874

facade (n.)

"front of a building,"1650s, from French *façade*

ARROWS ARE INTERFACES

- We pay too much attention to the boxes, not enough to the arrows.
- Every place an arrow crosses a boundary, we have an interface



FOR EACH ROLE

- Identify the constituents
- Find their needs, create use cases
- Determine how their needs get prioritized

FOR EACH API

- Discover throughput & latency requirements
- Decide on synchronicity
- Decide transport, framing, semantics
- Define test harness needs

ID	Name	Initiation	Sync?	Frequency	Size per	Transport	Framing	Encoding	Semantics
IF1	F5 Commands	Outbound	Sync	1 / min	5 KB	HTTP	XML	UTF-8	F5 Big IP
IF2	Healthcheck	Outbound	Sync	12 / min	1 KB	HTTP	JSON	UTF-8	Doc
IF3	Log aggregation	Inbound	Sync	12 / min	100 KB	SSH	Syslog	ASCII	Doc

ACTIVITY: INTERFACE TABLE

Create an interface table for our sample system:

- Outbound payment processor
- Outbound email, direct
- Outbound email, bulk

ARCHITECTURE QUALITIES

ARCHITECTURE QUALITIES

A.k.a. "the –ilities"

Sometimes called "non-functional requirements"

QUALITIES OBSERVED AT RUN-TIME

- Performance
- Security
- Availability
- Usability

From “Software Architecture in Practice”, Bass, Clements, Kazman

QUALITIES **NOT** OBSERVED AT RUN-TIME

- Scalability
- Modifiability
- Portability
- Integrability
- Reusability
- Testability

From “Software Architecture in Practice”, Bass, Clements, Kazman

There can be only one

top priority.

RANK THE QUALITIES

- Important to guide trade-offs later
- Very important that this isn’t just the tech team ranking these
- Hard discussions uncover assumptions

ACTIVITY: RANK THE QUALITIES

For our sample system, what are the most important qualities?

Write your top three, ranked from 1 – 3.

You are making a decision here, not taking a quiz.

But be sure you can make the case for your choice!

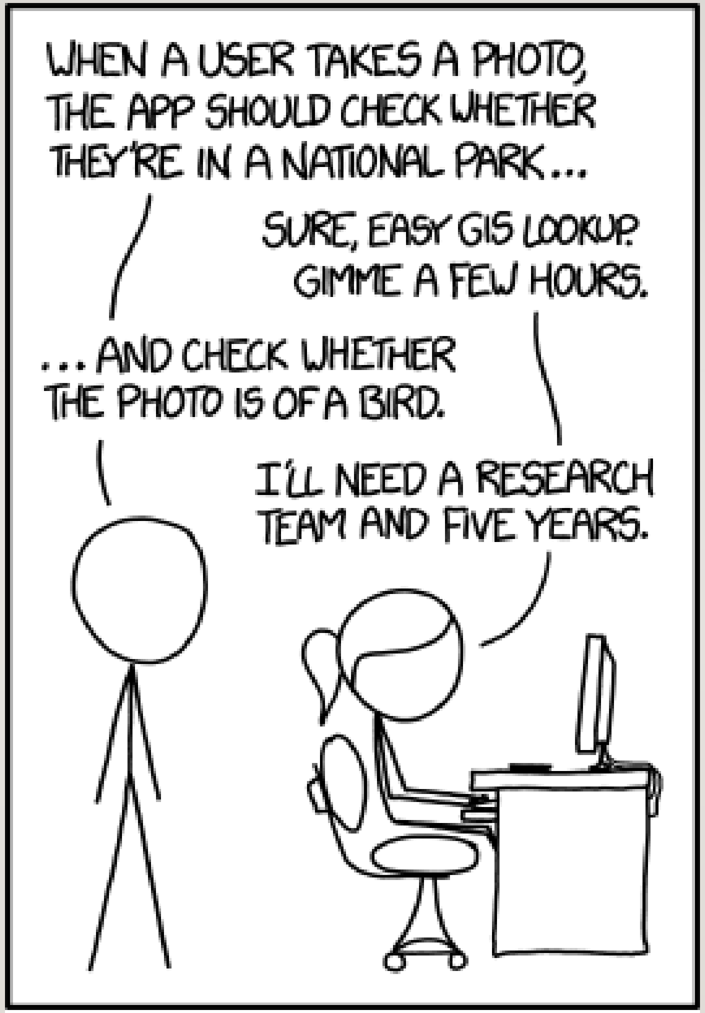
MVP, OFR, & TECH DEBT

Sometimes, "tech debt" just means architecture priorities have changed.

During	Emphasize
Before MVP	Modifiability
Finding Product/Market Fit	Scalability, Modifiability
Ramping Up	Security, Availability, Scalability

ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

ASRs

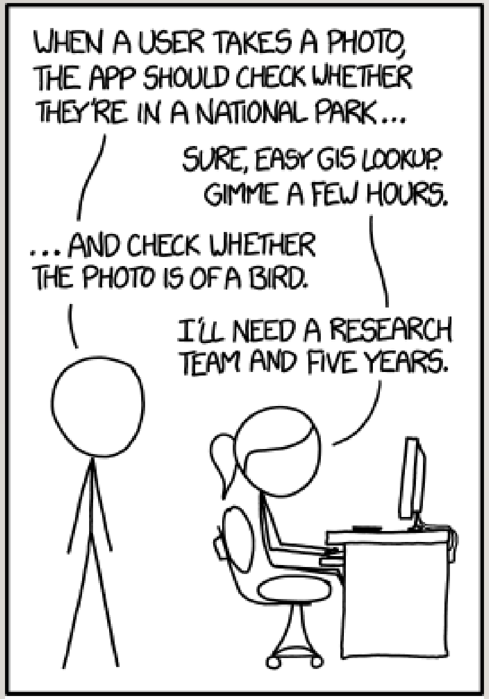

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

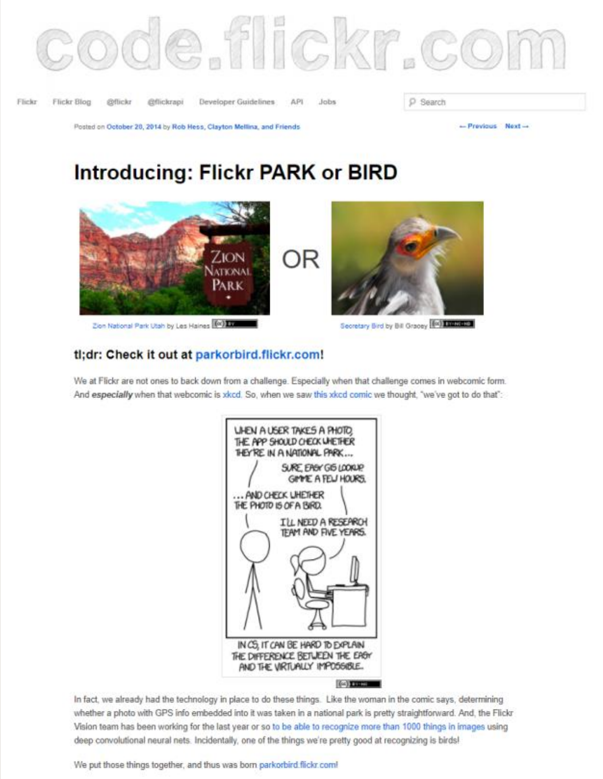
https://xkcd.com/1425/
Sept, 2014

Sept, 2014

Oct, 2014


WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.


code.flickr.com
Flickr
Flickr Blog
@flickr
@flickrapi
Developer Guidelines
API
Jobs
Search
← Previous
Next →
OR
ZION NATIONAL PARK
WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Posted on October 20, 2014 by Rob Hess, Clayton Mellina, and Friends

Introducing: Flickr PARK or BIRD

Zion National Park Utah by Les Haines

Secretary Bird by Bill Gracey

tl;dr: Check it out at parkorbird.flickr.com!

We at Flickr are not ones to back down from a challenge. Especially when that challenge comes in webcomic form. And *especially* when that webcomic is xkcd. So, when we saw this xkcd comic we thought, "we've got to do that":

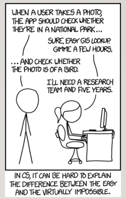
In fact, we already had the technology in place to do these things. Like the woman in the comic says, determining whether a photo with GPS info embedded into it was taken in a national park is pretty straightforward. And, the Flickr Vision team has been working for the last year or so to be able to recognize more than 1000 things in images using deep convolutional neural nets. Incidentally, one of the things we're pretty good at recognizing is birds!

We put those things together, and thus was born parkorbird.flickr.com!

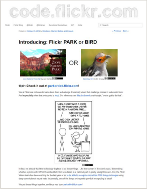
Feb, 2004

flickr

Sept, 2014


WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Oct, 2014



Posted on October 20, 2014 by Rob Hess, Clayton Mellina, and Friends

Introducing: Flickr PARK or BIRD

Zion National Park Utah by Les Haines

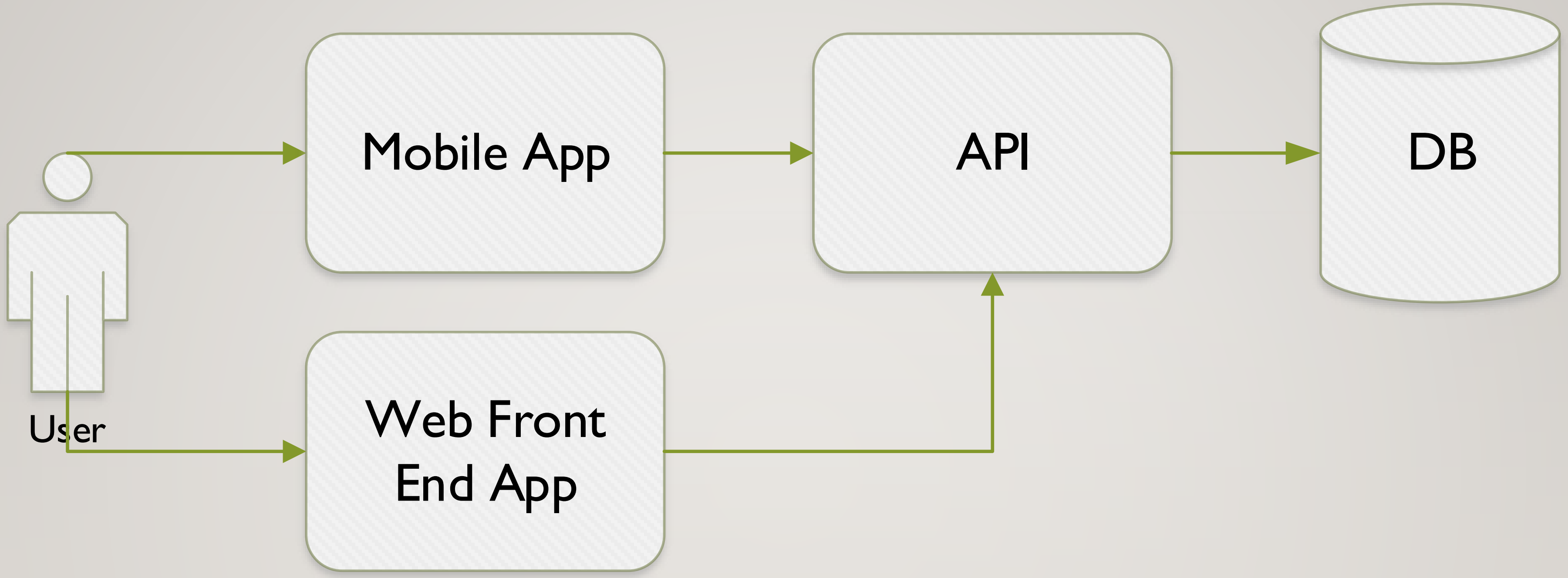
Secretary Bird by Bill Gracey

tl;dr: Check it out at parkorbird.flickr.com!

We at Flickr are not ones to back down from a challenge. Especially when that challenge comes in webcomic form. And **especially** when that webcomic is xkcd. So, when we saw this xkcd comic we thought, "we're got to do that":

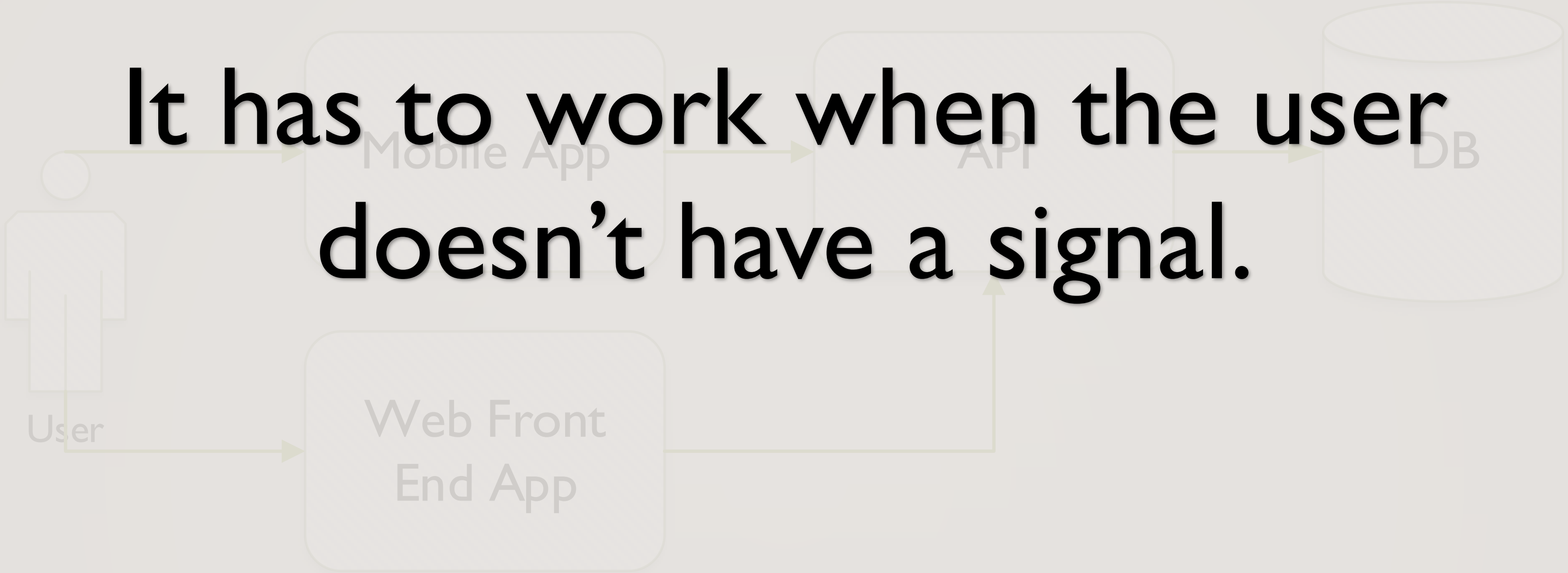
In fact, we already had the technology in place to do these things. Like the woman in the comic says, determining whether a photo with GPS info embedded into it was taken in a national park is pretty straightforward. And, the Flickr Vision team has been working for the last year or so to be able to recognize more than 1000 things in images using deep convolutional neural nets. Incidentally, one of the things we're pretty good at recognizing is birds!

We put these things together, and thus was born parkorbird.flickr.com!



EXAMPLE: MOBILE + WEB APP

It has to work when the user doesn't have a signal.



EXAMPLE: MOBILE + WEB APP

ASR

- Dramatically alters the architecture
- Causes addition of new “moving parts”
- Without it, the system will fail

“Architecture Killers”

HUNTING FOR ASRS

- Multiple languages, currencies, timezones
- Compliance & reporting
- Disconnected operation
- Any kind of database synchronization
- Active/active deployments
- Intelligence & semi-automated processes
- Customer service needs
- Human overrides
- High volume, low latency
- Strict serialization
- Interface with physical objects that move (especially when momentum is involved)
- Toxic, hazardous, radioactive, or remote environments

ASR OR NOT?

ID	Requirement	ASR?
R1	A vendor can set up a new service any time of day or night.	No
R2	A subscriber can upgrade a subscription any time, but can only downgrade at subscription renewal.	No
R3	A subscriber can pay with paper checks.	Yes – new role & GUI required
R4	A vendor can do their own customer service using our system.	Yes – new role, GUI, & new authn/authz
R5	A subscriber can call us to report a problem with service.	No
R6	A vendor can set up items in their own catalog service and they will appear on our site.	Yes – Data integration

CONSTRAINTS

CONSTRAINTS ARE MORE THAN REQUIREMENTS

- It maybe costly to break a requirement, but a constraint is absolute.
- If a constraint is broken, then the system **must not** go live.
- Constraints are imposed from external forces:
  - Law
  - Industry regulation
  - Stakeholders

ID	Headline	Originator	Local Expert	Brief Description	More Detail
C1	GDPR	EU Law	Niles Summerbottom	Limits on collection & storage of PII. “Right to be forgotten”	Doc
C2	CAN SPAM	US 16 CFR Part 316	Ricky Bobby	Restrictions and requirements on email	Doc
C3	PCI DSS	Payment Card Industry	Scrooge McDuck	Restrictions on handling credit card data	Doc
C4	HIPAA	…			
C5	Peppa	…			
…					

CONSTRAINTS ARE LESS UNIQUE THAN REQUIREMENTS

- We will find similar constraints when we look at systems in the same:
  - Domain
  - Market
  - Company

- Much of the guidance can be shared from one system to the next.
- But it’s usually left as tacit knowledge.

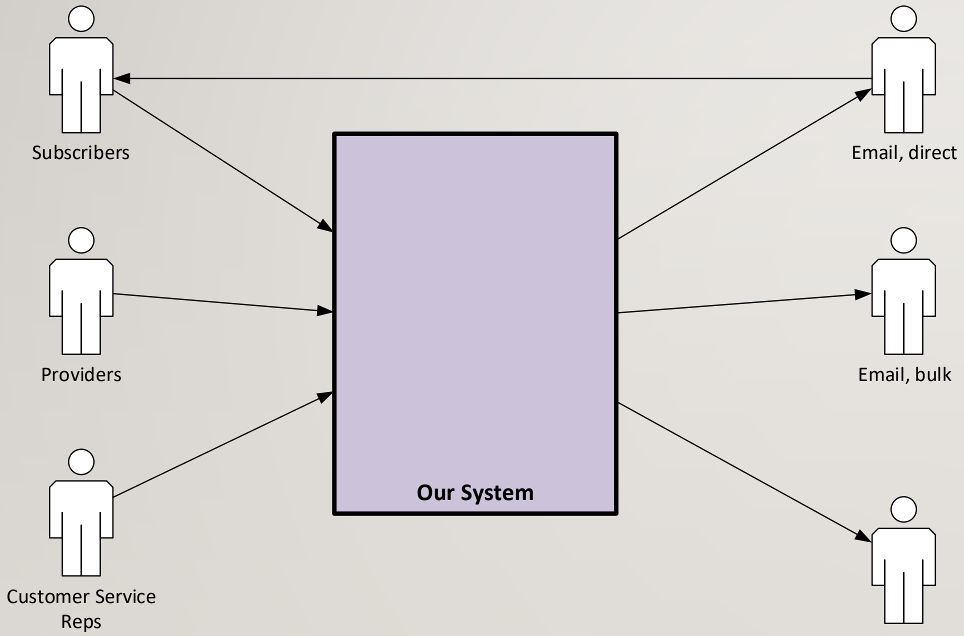
ACTIVITY: CONSTRAINTS IN YOUR LOCALE?

- What are some constraints that our sample system would face here?

CHECKPOINT

Where are we now?

THE STORY SO FAR



Context Diagram

- ASR
  - White-label with custom domains
- Constraints
  - CAN SPAM, PCI DSS, GDPR
- Architecture qualities
  1. Availability
  2. Security
  3. Scalability

Time to open the box and go inside

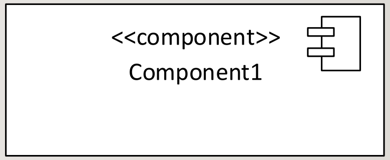
DECOMPOSING THE SYSTEM

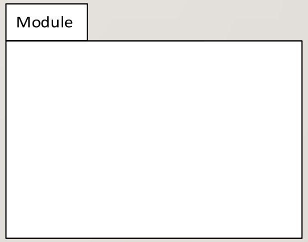
INTERNAL BOUNDARIES

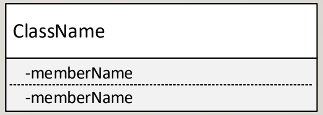
- Our decomposition *must*:
  - Terminate every external interface (human or system) at a component.
  - Deliver the features
- Our decomposition *should*:
  - Support the system-wide qualities
  - Allow independent development
  - Separate concerns into mechanisms
  - Isolate the effects of changes

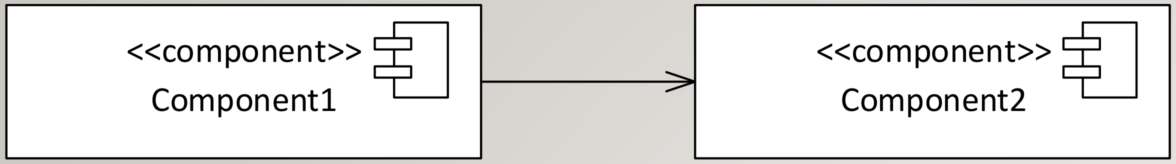
TERMINOLOGY & NOTATION

- Component
  - Part of the dynamic behavior
  - A runtime entity
  - Interacts with other components
- Module
  - Part of the static structure
  - Development/compile time entity

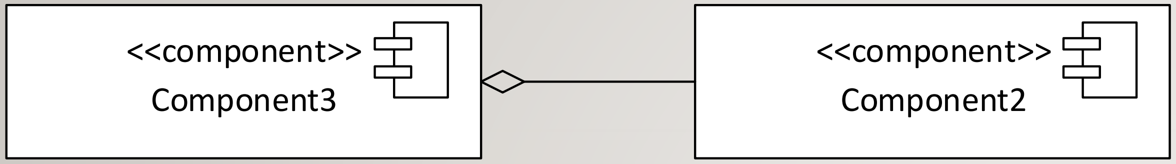




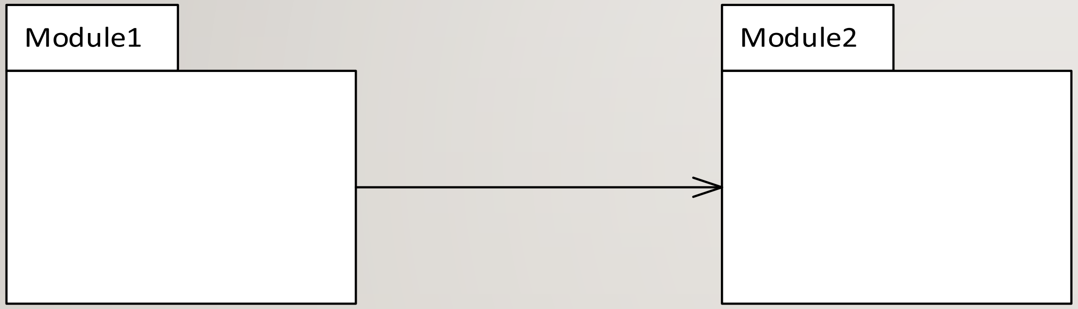




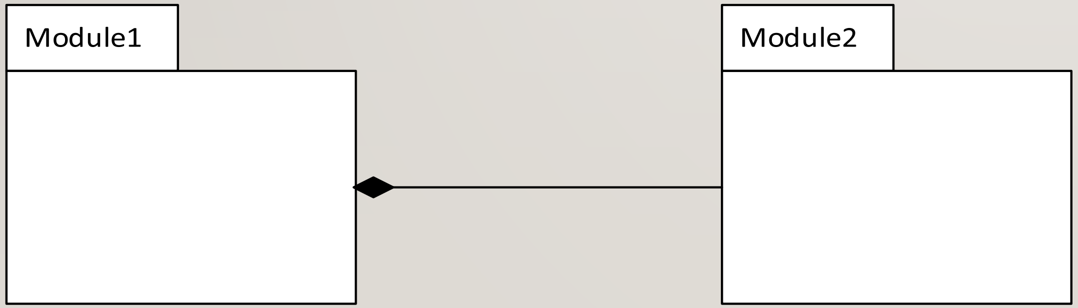
Statement about runtime behavior.
Component1 calls Component2



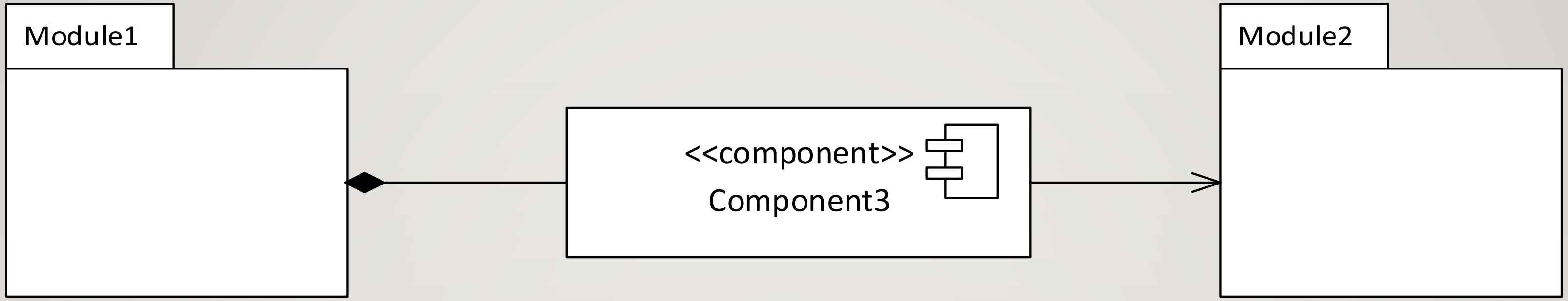
Statement about runtime behavior.
Component3 contains Component2



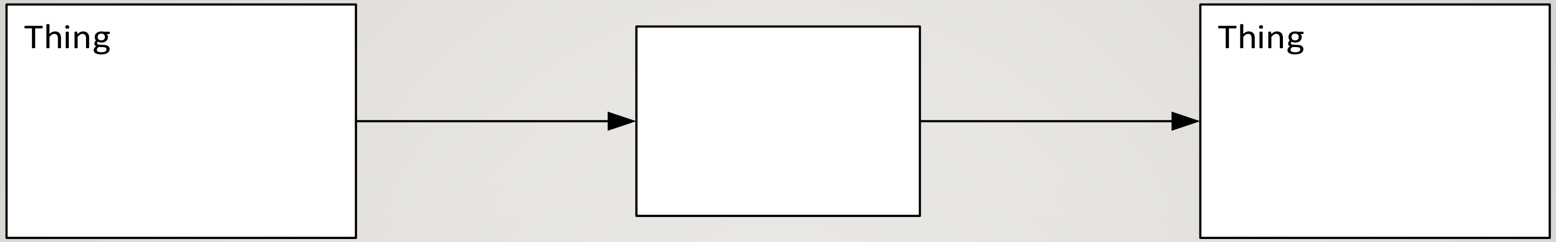
Statement about source code structure.
Module1 uses Module2.



Statement about source code structure.
Module1 encloses Module2.



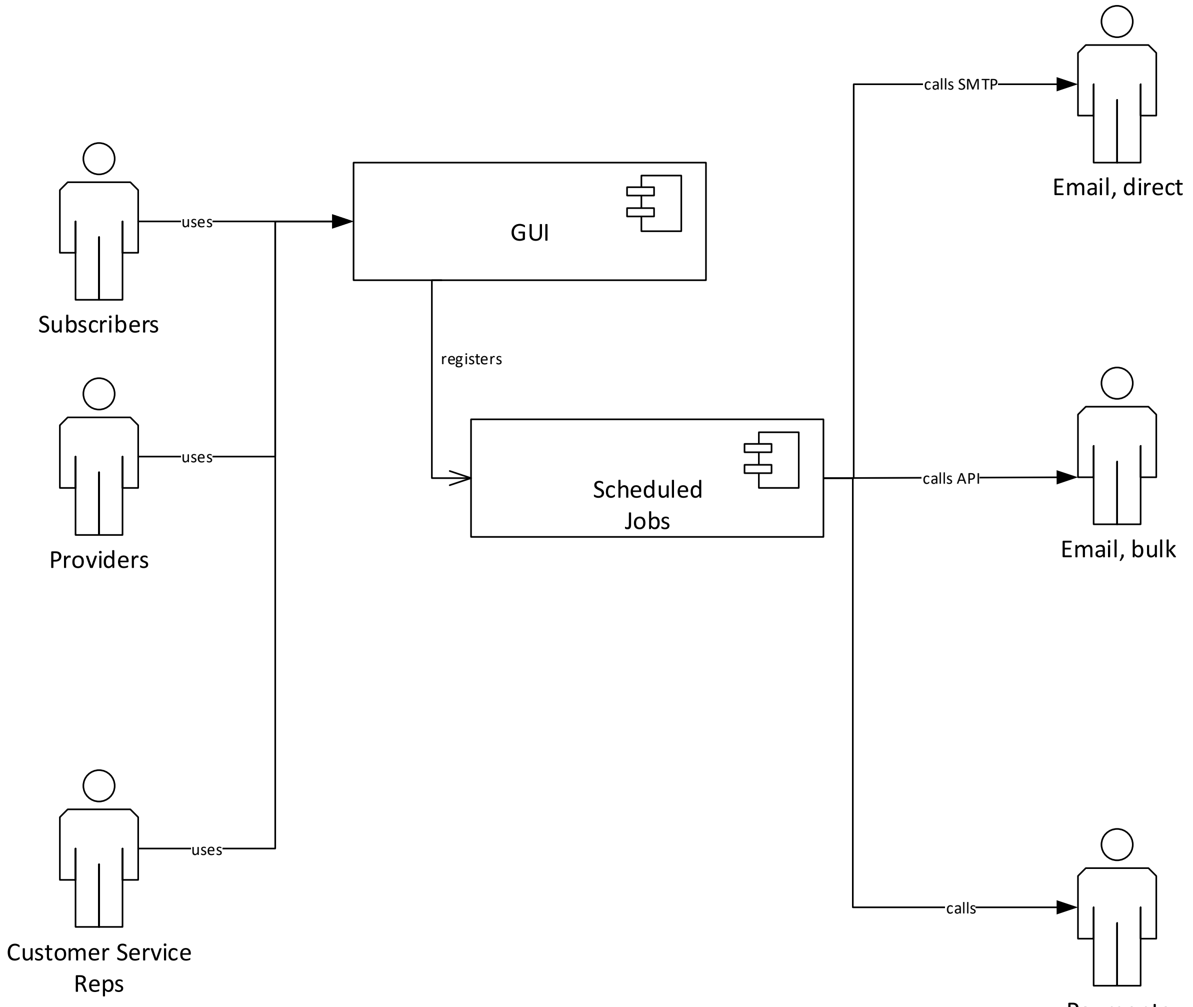
✗ Doesn’t look like anything to me.

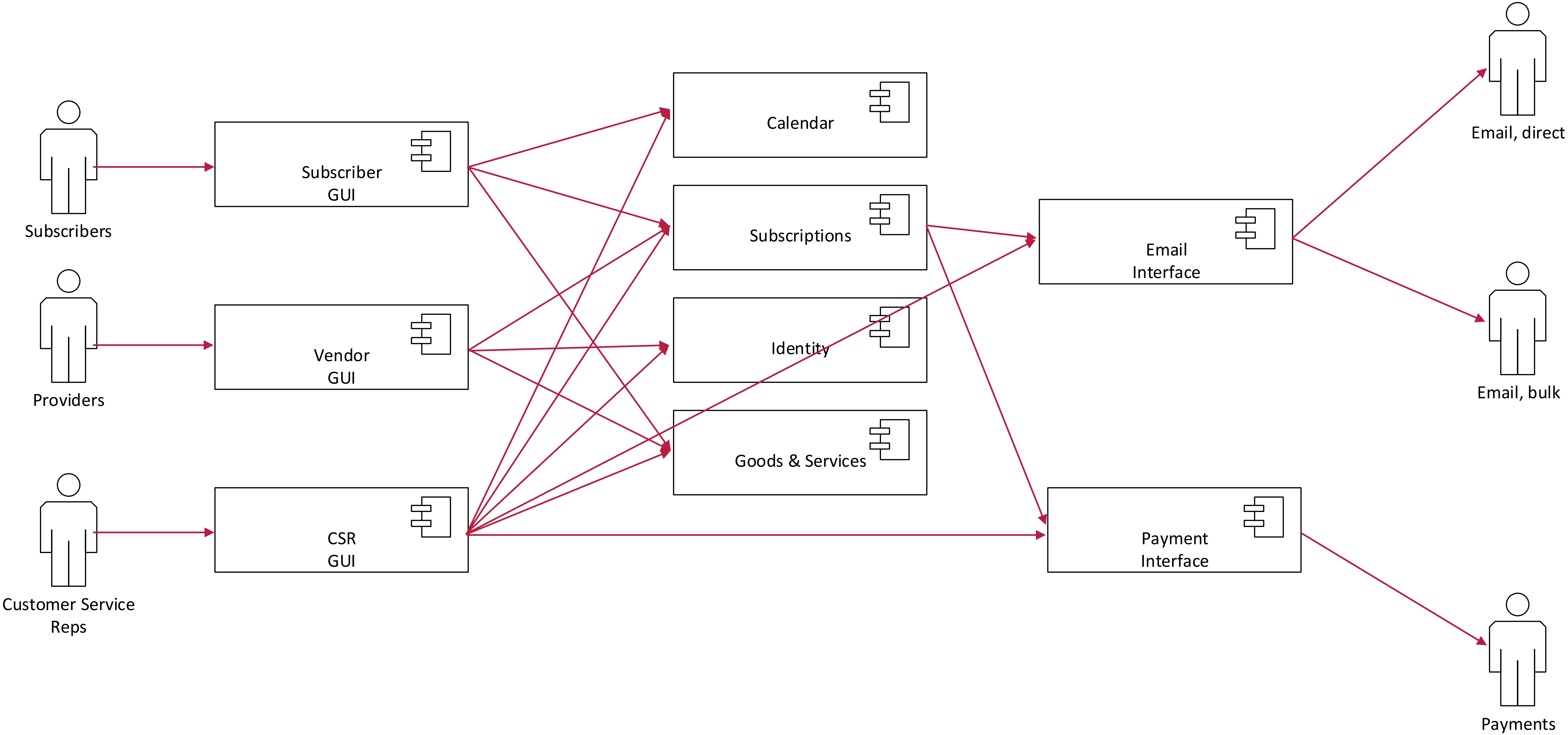


😟 I have no idea what this is supposed to mean.

MAXIMS

- Use both static and dynamic structures
  - Can this be a library?
  - Does this need to be a service?
- Look for common mechanisms you can factor out.
- Ask "what knowledge must be shared across this boundary?"


Subscribers
Providers
Customer Service Reps
uses
uses
uses
GUI
registers
Scheduled Jobs
calls SMTP
calls API
calls
Email, direct
Email, bulk
Payments


Subscribers
Providers
Customer Service Reps
Subscriber GUI
Vendor GUI
CSR GUI
Calendar
Subscriptions
Identity
Goods & Services
Email Interface
Payment Interface
Email, direct
Email, bulk
Payments

EVALUATE YOUR DECOMPOSITION

- “Test” your design by simulation.
- Assign people to play each component
- Pick a use case
- Pass a ball or token around to simulate control flow
- Walk through the use case and pass the token
- Score it like golf

INFORMATION HIDING

- David Parnas, “On the Criteria To Be Used in Decomposing Systems into Modules", Communications of the ACM, 1972
- KWIC Index – a permuted index
- Input – ordered lines, made up of ordered words, made up of ordered characters.
- Output – listing of all “circular shifts” of all lines, in alphabetic order

A KWIC EXAMPLE

- Input

```
Software comprises an endless supply of structures.
```

- Output

```
an endless supply of structures. Software comprises
comprises an endless supply of structures. Software
endless supply of structures. Software comprises an
of structures. Software comprises an endless supply
Software comprises an endless supply of structures.
structures. Software comprises an endless supply of
supply of structures. Software comprises an endless
```

MODULARIZATION 1

1. **Input**
   Read EBCDIC characters, store them in core. 6-bit characters packed 4 per word. EOL is a special character.
2. **Circular shifter**
   Prepare index; pair of addr of first char of shift, original index of line in input array
3. **Alphabetizer**
   Take arrays from 1 & 2, produce new array of pairs like in 2, but in alphabetical order.
4. **Output**
   Using arrays from 1 & 3, format output
5. **Control**
   Allocate memory, call operations in1 - 4, report errors.

ACTIVITY: EFFECT OF CHANGES

For each change case listed here, count how many modules have to be changed:

1. Read and print ASCII instead of EBCDIC.
2. Stop using packed characters, store one character per word.
3. Write index for circular shifts to offline storage instead of core to support larger input documents.
4. Read and write Unicode and UTF-8. Collate properly for Unicode in locale.

MODULARIZATION II

1. **Line Storage**
   Offers functional interface: SETCH, GETCH, GETW, DELW, DELLINE
2. **Input**
   Reads EBCDIC chars, calls line storage to put them into lines.
3. **Circular Shifter**
   Offers same interface as line storage. Makes it appear to have all shifts of all lines.
4. **Alphabetizer**
   Offers sort function INIT, and access function iTH that gets a line.
5. **Output**
   Repeatedly call iTH on alphabetizer, printing the line.
6. **Control**
   Similar to first approach, call each module in sequence.

ACTIVITY: EFFECT OF CHANGES

Let's evaluate the second modularization against the same change cases:

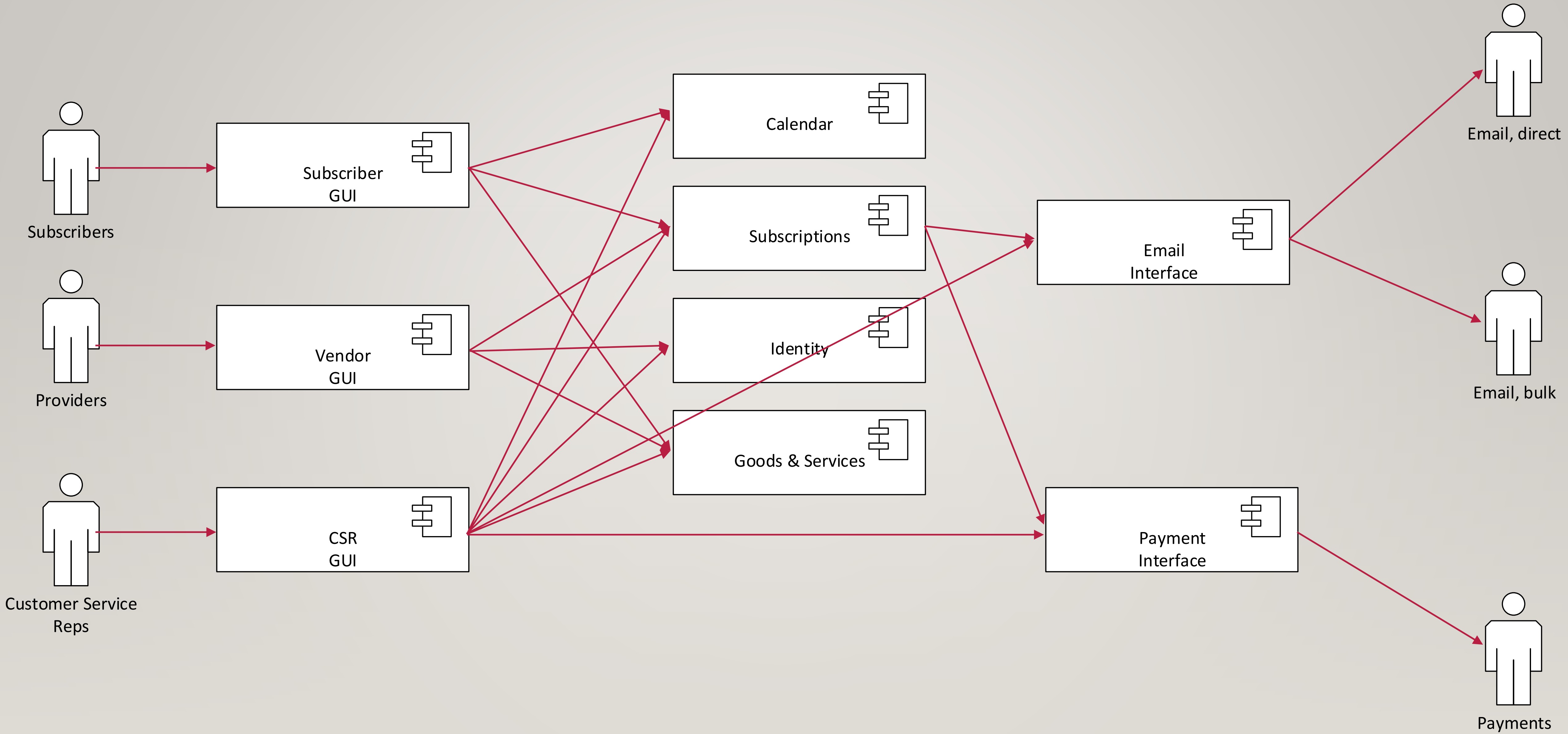
1. Read and print ASCII instead of EBCDIC.
2. Stop using packed characters, store one character per word.
3. Write index for circular shifts to offline storage instead of core to support larger input documents.
4. Read and write Unicode and UTF-8. Collate properly for Unicode in locale.

WHY IS THE SECOND ONE BETTER?

- It hides decisions inside modules.
- Functional interfaces provide an abstract representation of the underlying data.
- Information hiding

IN OUR SAMPLE SYSTEM

- We need to decompose the interior
    - Allow multiple people to work
    - Work toward our system-wide priorities


Subscribers
Providers
Customer Service Reps
Subscriber GUI
Vendor GUI
CSR GUI
Calendar
Subscriptions
Identity
Goods & Services
Email Interface
Payment Interface
Email, direct
Email, bulk
Payments

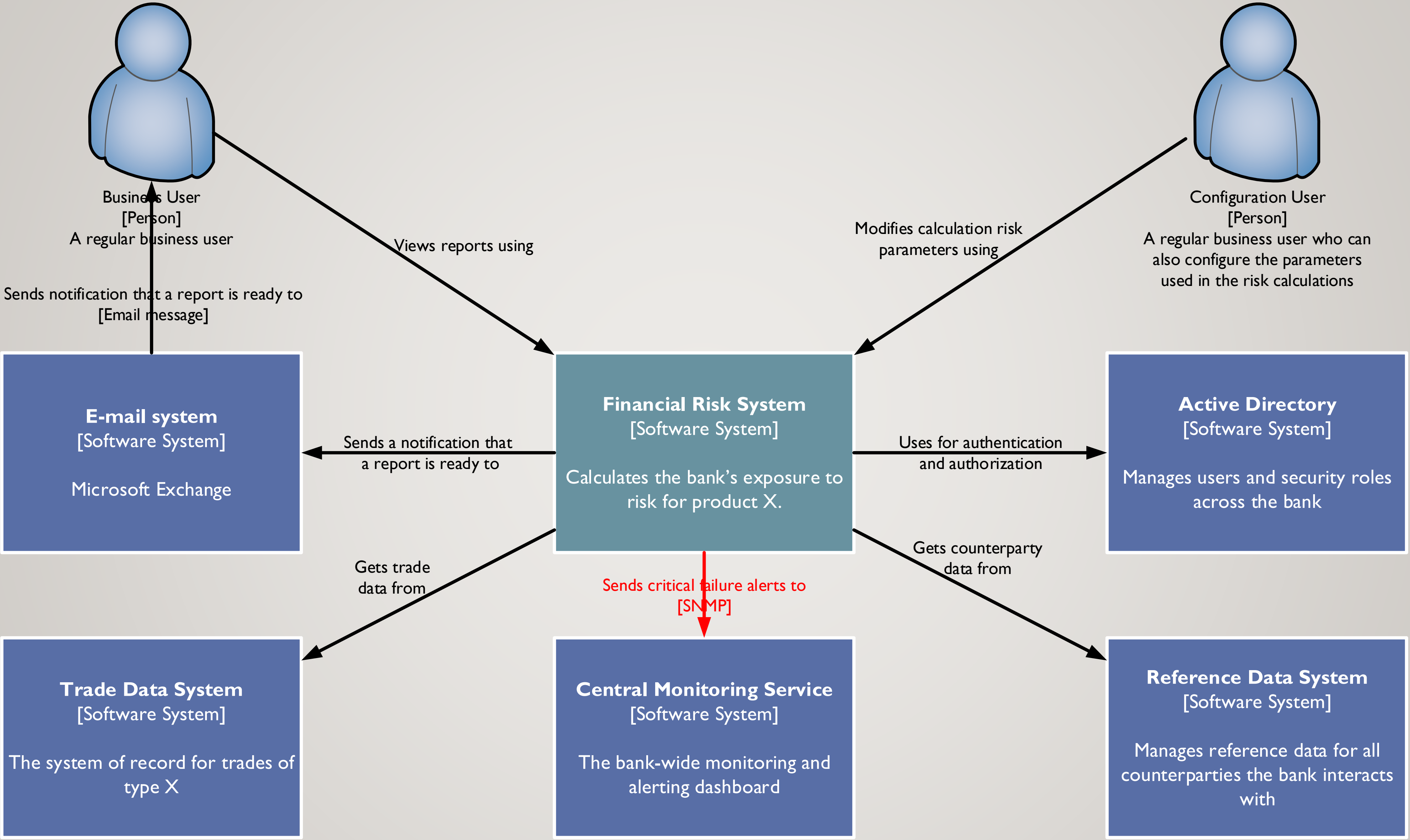
HOMEWORK

- Find a better decomposition for our subscription system.
- Take advantage of components & modules.
  - Think about libraries that would be needed in more than one place.
  - Think about components that would be useful *on their own*
- Component ≠ microservice
- This decomposition should be independent of deployment decisions.
- **Hint**: If you can solve the "CSR problem" you're way ahead.
- **"Extra Credit"**: Compute the Modularization Quality (MQ) for different structures.

VIEWS YOU CAN USE

C4 – Context, Containers, Components, Classes

CONTEXT DIAGRAM



Business User
[Person]
A regular business user

Configuration User
[Person]
A regular business user who can also configure the parameters used in the risk calculations

Views reports using

Modifies calculation risk parameters using

Sends notification that a report is ready to
[Email message]

Financial Risk System
[Software System]
Calculates the bank’s exposure to risk for product X.

E-mail system
[Software System]
Microsoft Exchange

Sends a notification that a report is ready to

Active Directory
[Software System]
Manages users and security roles across the bank

Uses for authentication and authorization

Gets trade data from

Sends critical failure alerts to
[SNMP]

Gets counterparty data from

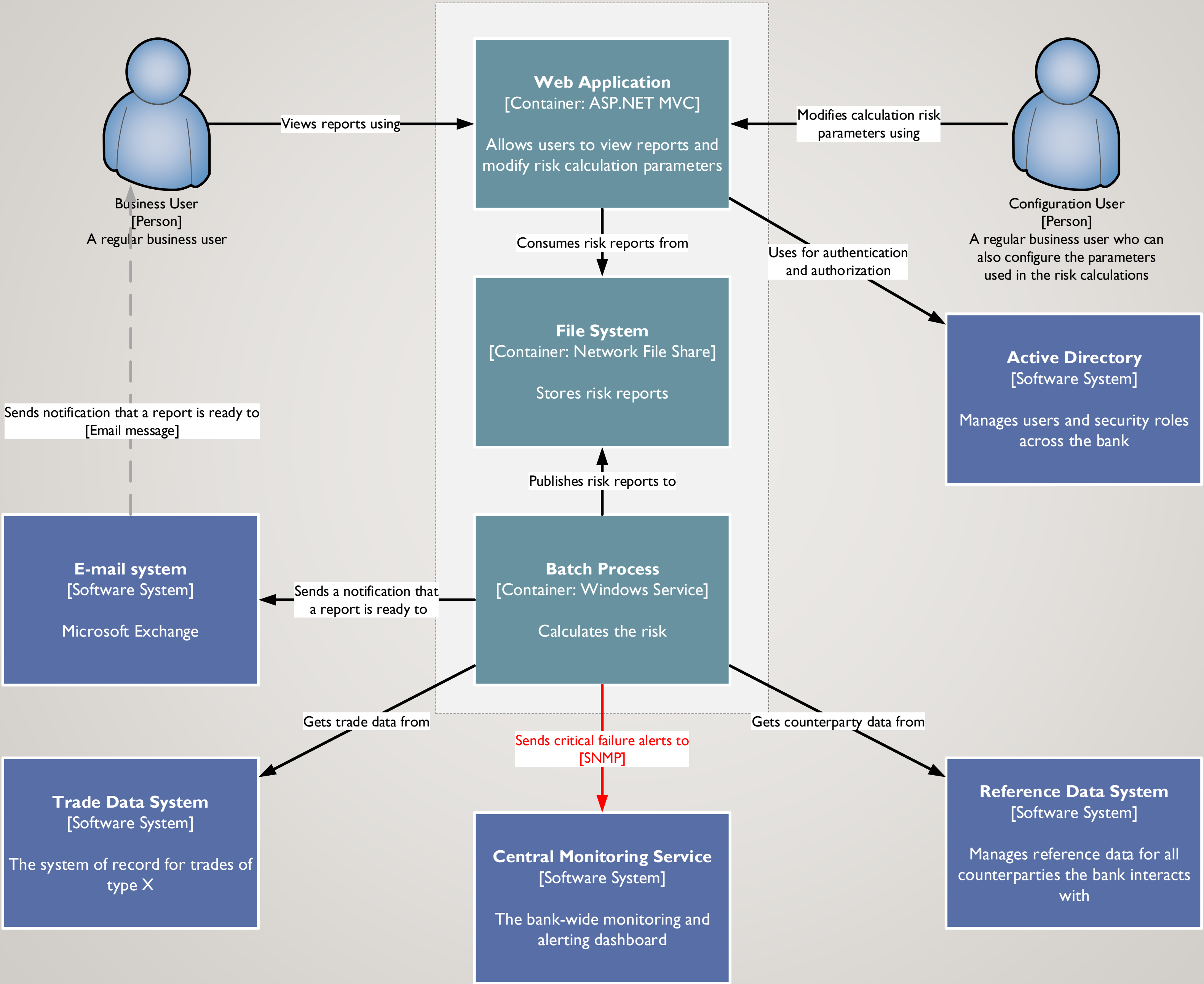
Trade Data System
[Software System]
The system of record for trades of type X

Central Monitoring Service
[Software System]
The bank-wide monitoring and alerting dashboard

Reference Data System
[Software System]
Manages reference data for all counterparties the bank interacts with

http://c4model.com/

CONTAINER DIAGRAM



Business User
[Person]
A regular business user

Configuration User
[Person]
A regular business user who can also configure the parameters used in the risk calculations

Web Application
[Container: ASP.NET MVC]
Allows users to view reports and modify risk calculation parameters

File System
[Container: Network File Share]
Stores risk reports

Batch Process
[Container: Windows Service]
Calculates the risk

Active Directory
[Software System]
Manages users and security roles across the bank

E-mail system
[Software System]
Microsoft Exchange

Trade Data System
[Software System]
The system of record for trades of type X

Central Monitoring Service
[Software System]
The bank-wide monitoring and alerting dashboard

Reference Data System
[Software System]
Manages reference data for all counterparties the bank interacts with

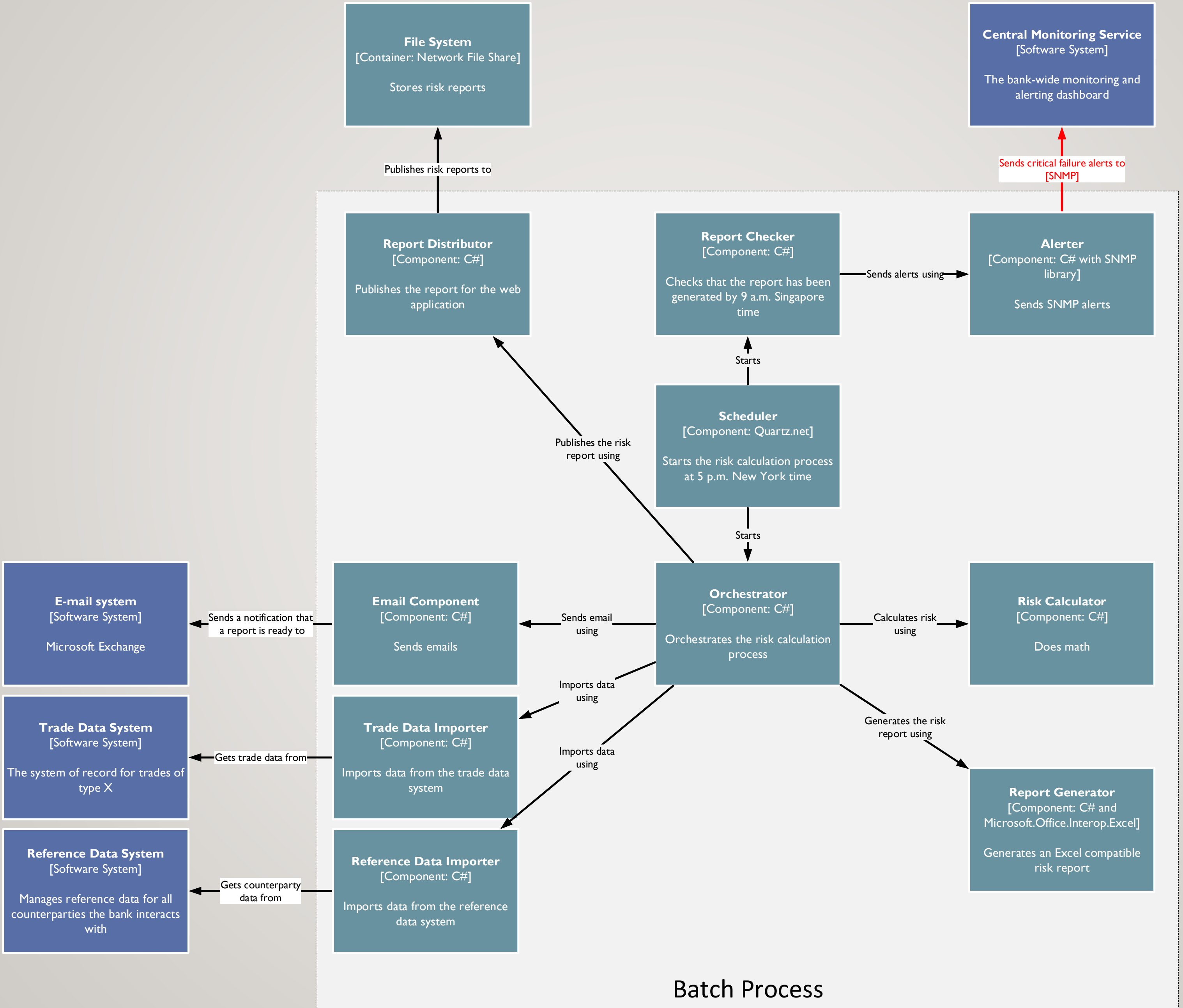
Views reports using
Modifies calculation risk parameters using
Consumes risk reports from
Uses for authentication and authorization
Sends notification that a report is ready to [Email message]
Publishes risk reports to
Sends a notification that a report is ready to
Gets trade data from
Sends critical failure alerts to [SNMP]
Gets counterparty data from

http://c4model.com/

ACTIVITY: MEMORY CHECK

- What was the difference between a “Component” and a “Module”?

COMPONENT DIAGRAM



File System
[Container: Network File Share]

Stores risk reports

Central Monitoring Service
[Software System]

The bank-wide monitoring and alerting dashboard

Publishes risk reports to

Sends critical failure alerts to [SNMP]

Report Distributor
[Component: C#]

Publishes the report for the web application

Report Checker
[Component: C#]

Checks that the report has been generated by 9 a.m. Singapore time

Sends alerts using

Alerter
[Component: C# with SNMP library]

Sends SNMP alerts

Starts

Scheduler
[Component: Quartz.net]

Starts the risk calculation process at 5 p.m. New York time

Publishes the risk report using

Starts

E-mail system
[Software System]

Microsoft Exchange

Sends a notification that a report is ready to

Email Component
[Component: C#]

Sends emails

Sends email using

Orchestrator
[Component: C#]

Orchestrates the risk calculation process

Calculates risk using

Risk Calculator
[Component: C#]

Does math

Imports data using

Imports data using

Generates the risk report using

Trade Data System
[Software System]

The system of record for trades of type X

Gets trade data from

Trade Data Importer
[Component: C#]

Imports data from the trade data system

Report Generator
[Component: C# and Microsoft.Office.Interop.Excel]

Generates an Excel compatible risk report

Reference Data System
[Software System]

Manages reference data for all counterparties the bank interacts with

Gets counterparty data from

Reference Data Importer
[Component: C#]

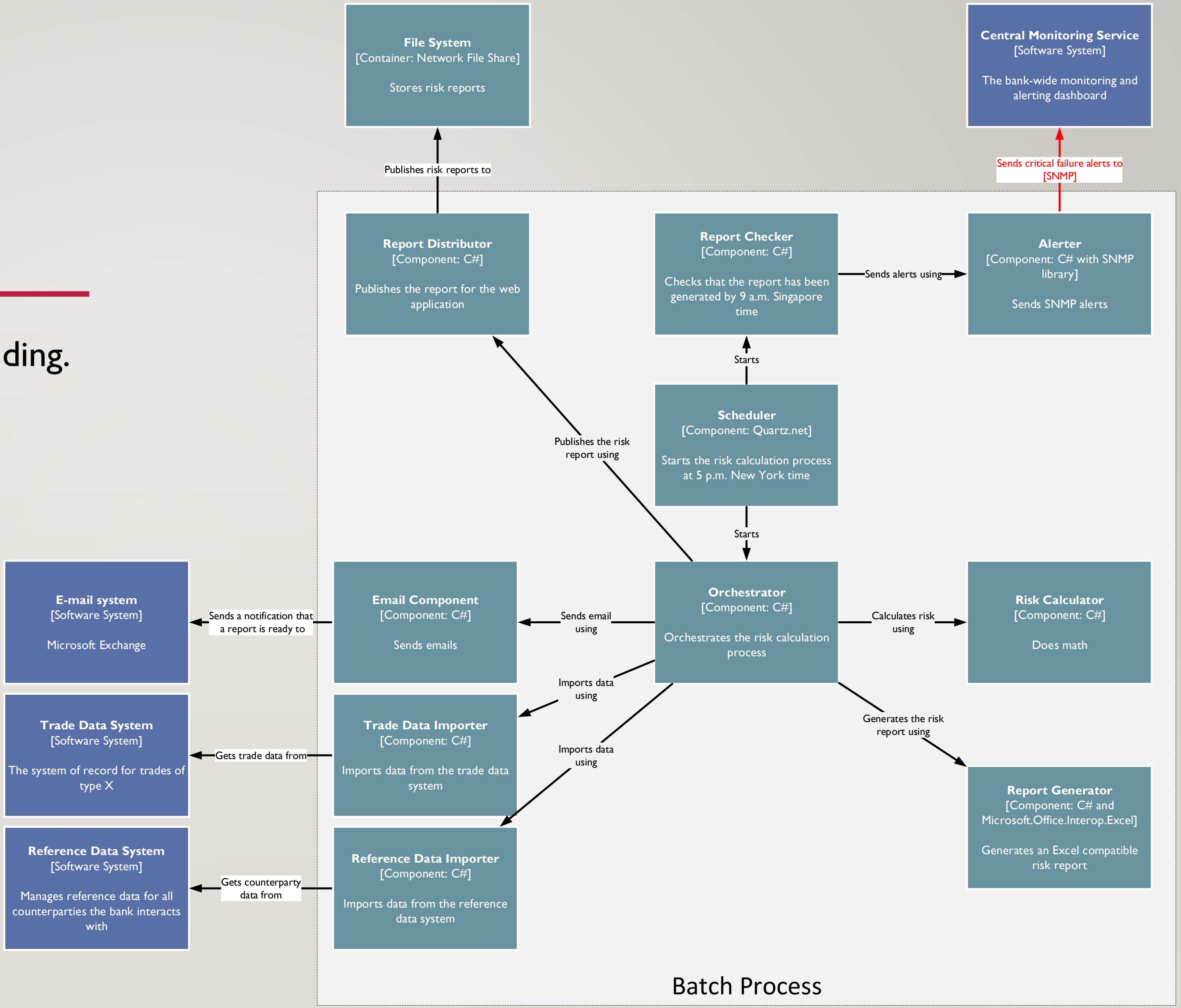
Imports data from the reference data system

Batch Process

http://c4model.com/

COMPONENT DIAGRAM

- Remember the principle of decision hiding.
- How do you think this decomposition does at that?
- What decisions appear to be exposed across boundaries here?



File System
[Container: Network File Share]
Stores risk reports

Central Monitoring Service
[Software System]
The bank-wide monitoring and alerting dashboard

Publishes risk reports to

Sends critical failure alerts to [SNMP]

Report Distributor
[Component: C#]
Publishes the report for the web application

Report Checker
[Component: C#]
Checks that the report has been generated by 9 a.m. Singapore time

Sends alerts using

Alerter
[Component: C# with SNMP library]
Sends SNMP alerts

Starts

Scheduler
[Component: Quartz.net]
Starts the risk calculation process at 5 p.m. New York time

Publishes the risk report using

Starts

E-mail system
[Software System]
Microsoft Exchange

Sends a notification that a report is ready to

Email Component
[Component: C#]
Sends emails

Sends email using

Orchestrator
[Component: C#]
Orchestrates the risk calculation process

Calculates risk using

Risk Calculator
[Component: C#]
Does math

Imports data using

Imports data using

Generates the risk report using

Trade Data System
[Software System]
The system of record for trades of type X

Gets trade data from

Trade Data Importer
[Component: C#]
Imports data from the trade data system

Report Generator
[Component: C# and Microsoft.Office.Interop.Excel]
Generates an Excel compatible risk report

Reference Data System
[Software System]
Manages reference data for all counterparties the bank interacts with

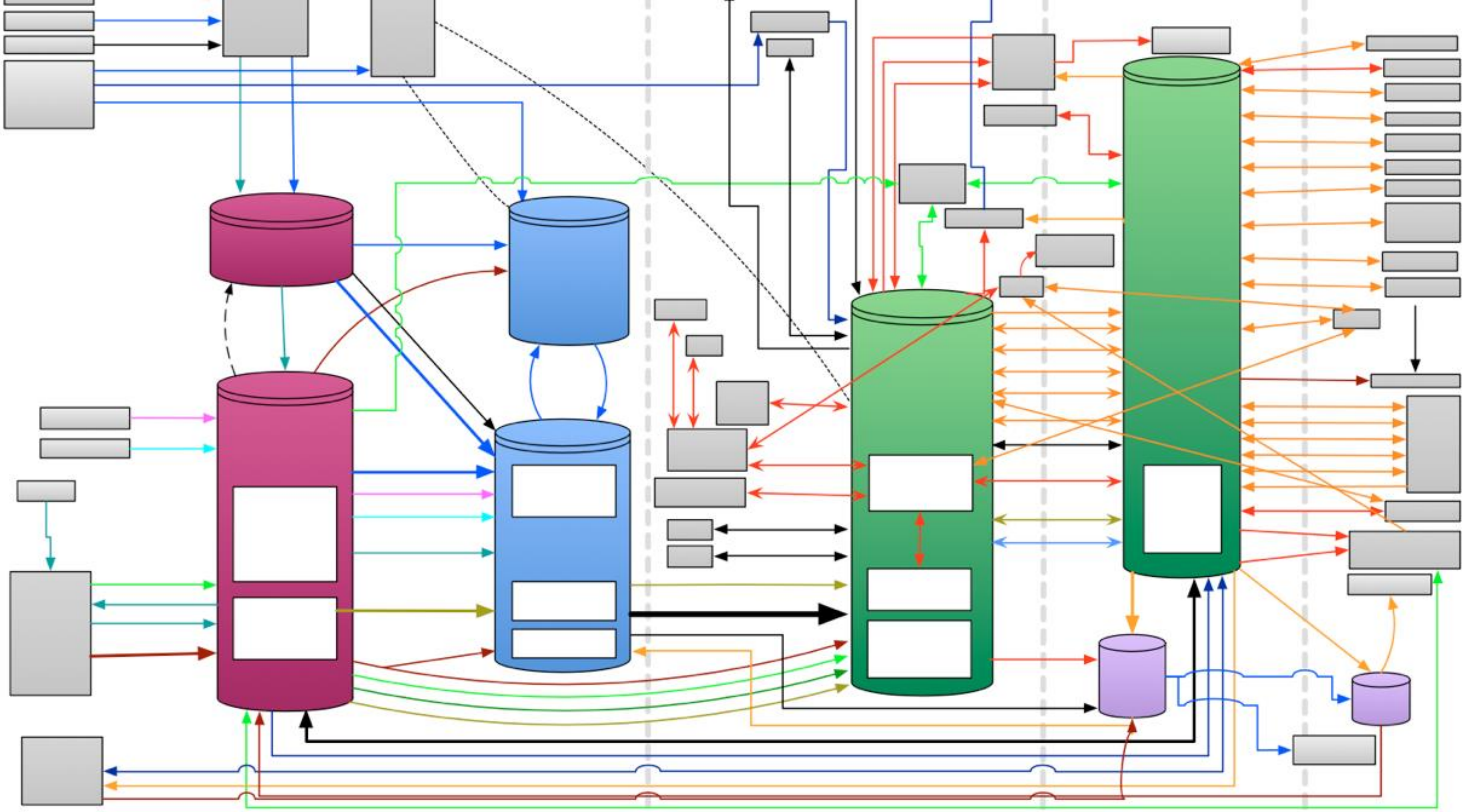
Gets counterparty data from

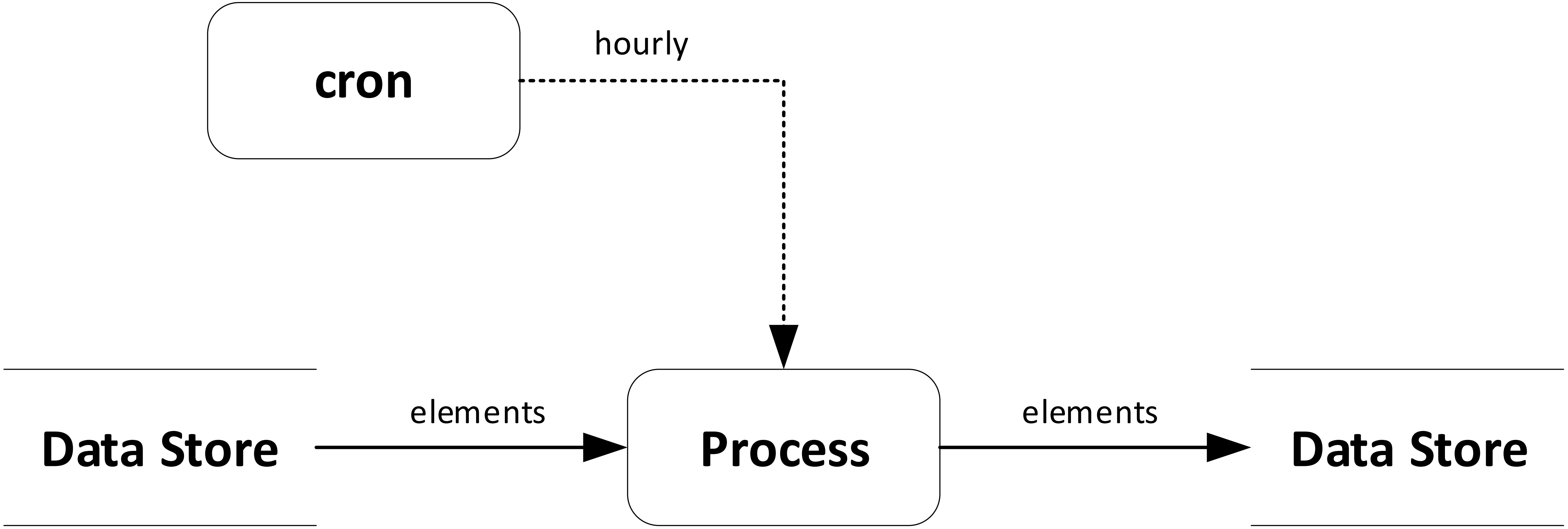
Reference Data Importer
[Component: C#]
Imports data from the reference data system

Batch Process

VIEWS YOU CAN USE

Data Flow Diagram





cron

hourly

Data Store

elements

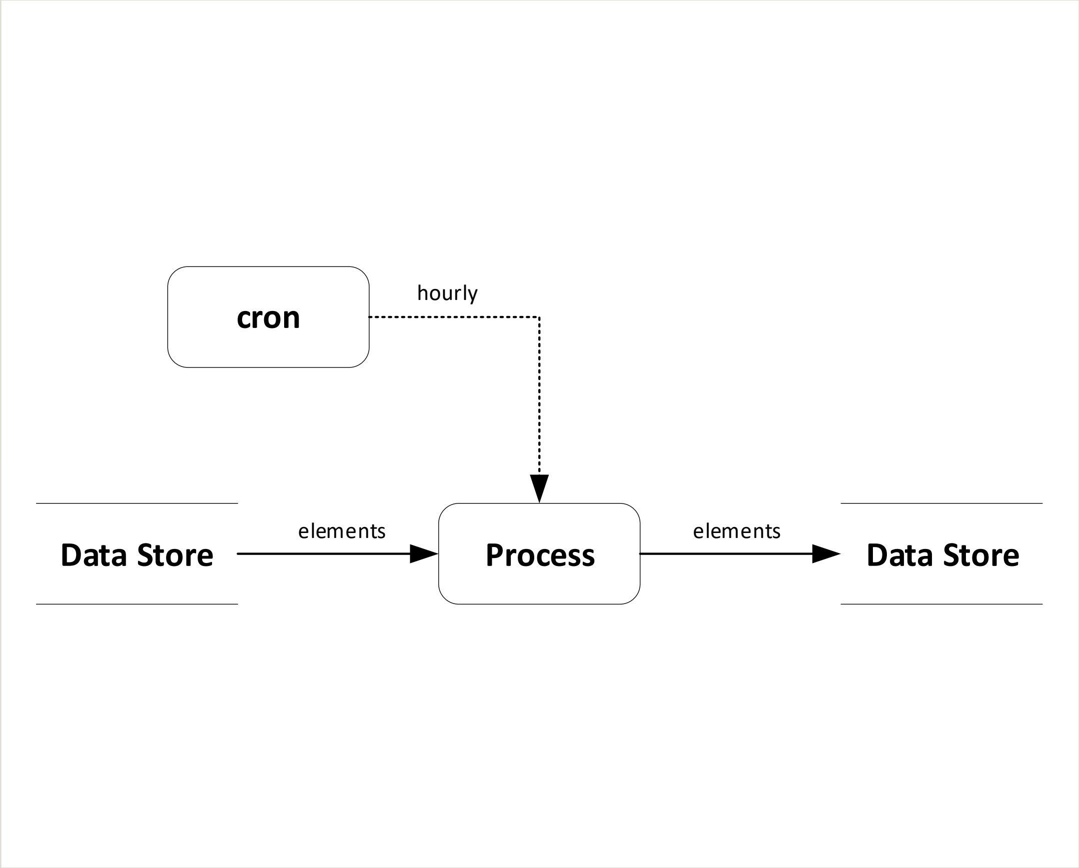
Process

elements

Data Store

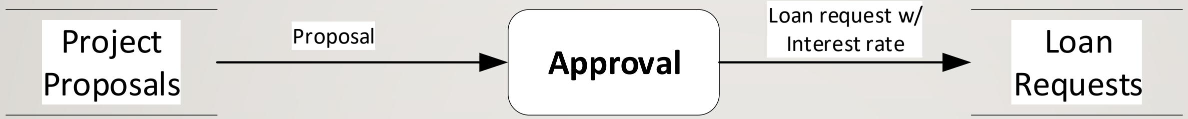
RULES FOR DFDs

1. Data never flows directly from one store to another.
2. Processes receive, transform, and transmit data.
3. Processes are triggered by a control flow.
4. Arrows show the direction data goes, not the kind of call.



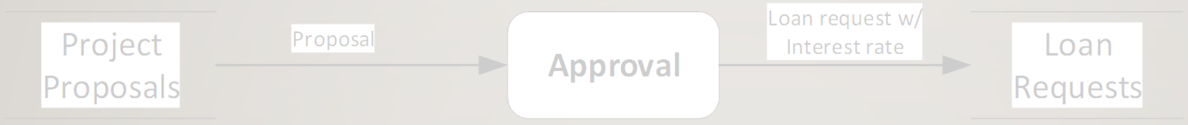
WHAT'S WRONG WITH EACH OF THESE?

A

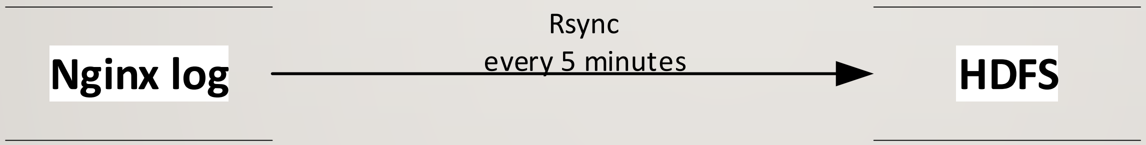


WHAT'S WRONG WITH EACH OF THESE?

A

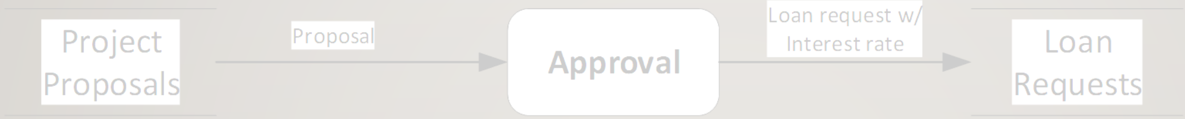


B



WHAT'S WRONG WITH EACH OF THESE?

A



Project Proposals

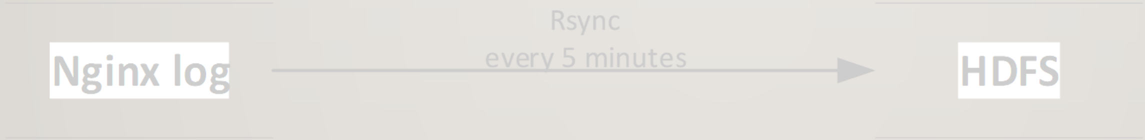
Proposal

Approval

Loan request w/ Interest rate

Loan Requests

B

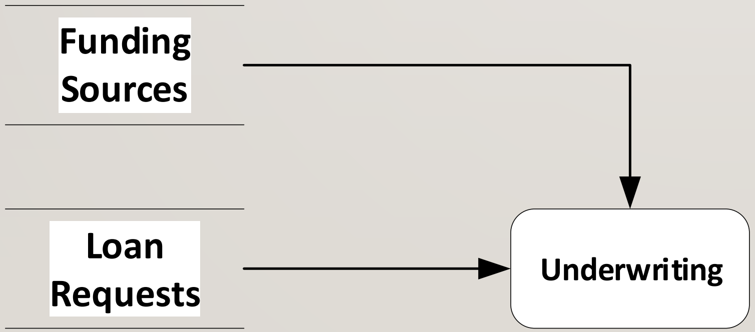


Nginx log

Rsync every 5 minutes

HDFS

C

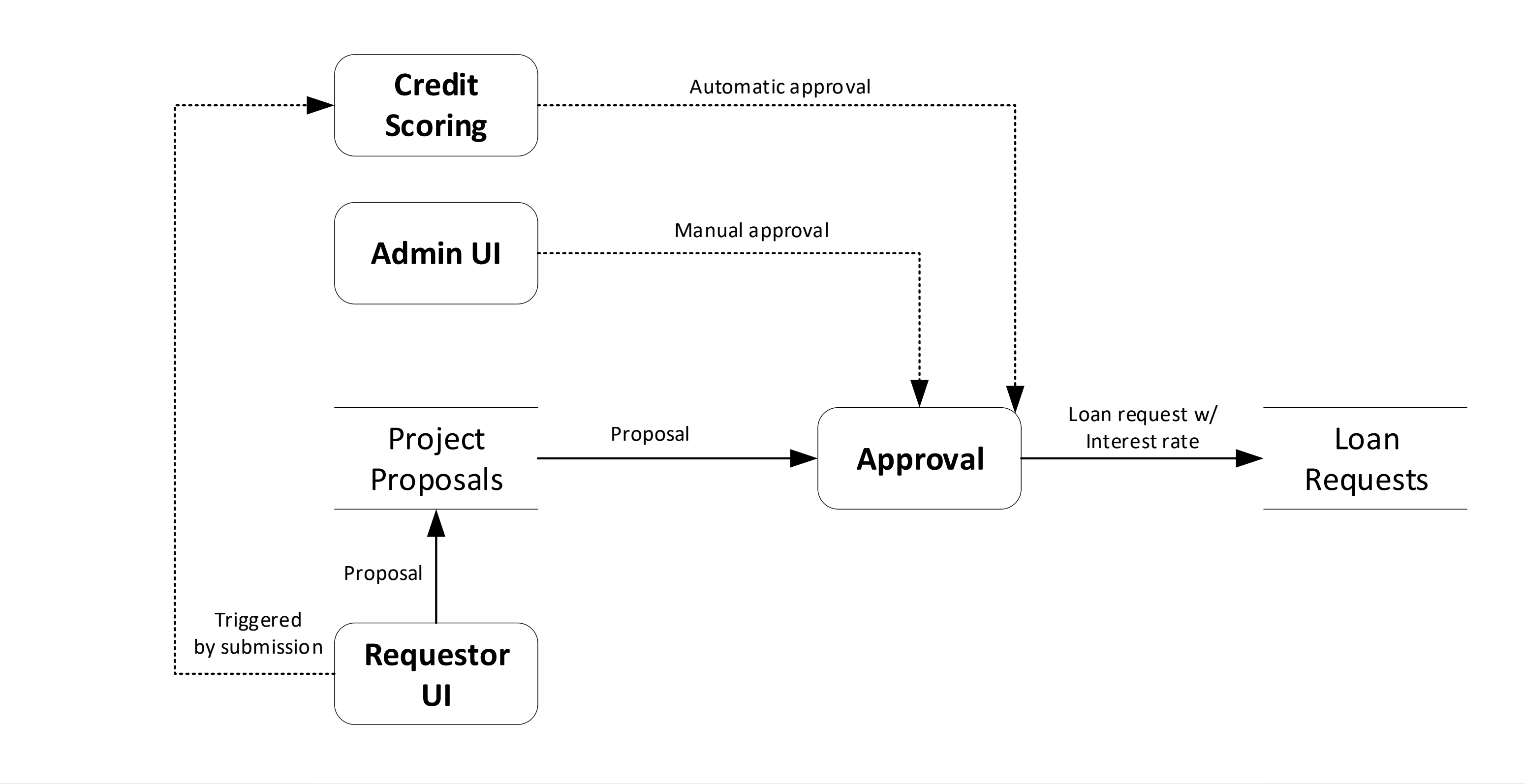


Funding Sources

Loan Requests

Underwriting

EXAMPLE FROM A PROJECT



NEXT UP: INCREMENTAL ARCHITECTURE