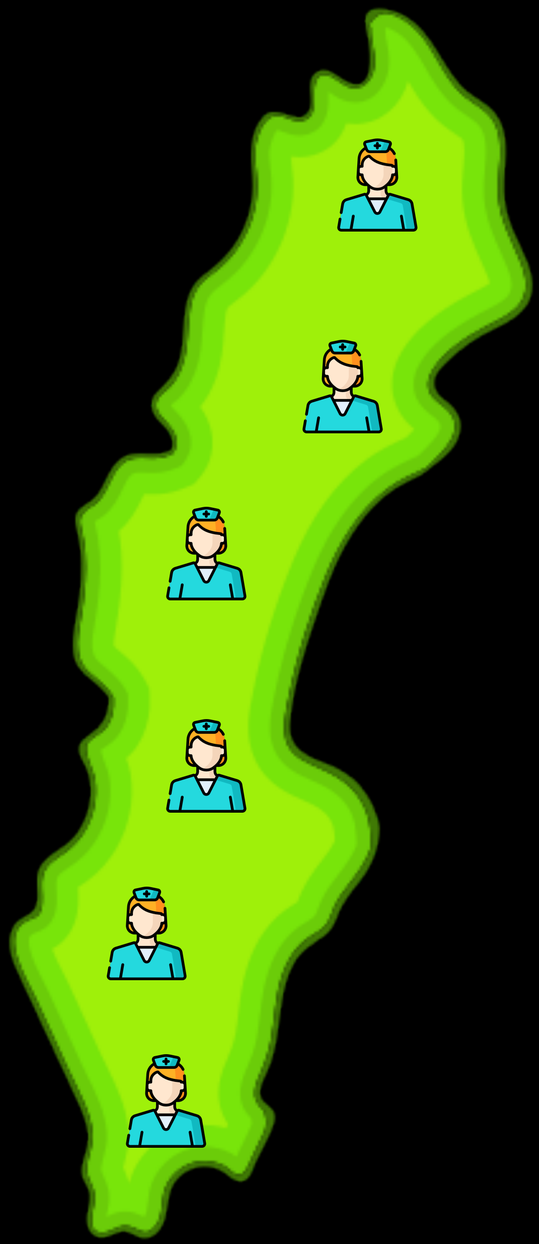
DESIGNING FOR HIGH AVAILABILITY IN A DISTRIBUTED ARCHITECTURE

JOHAN ZETTERSTRÖM

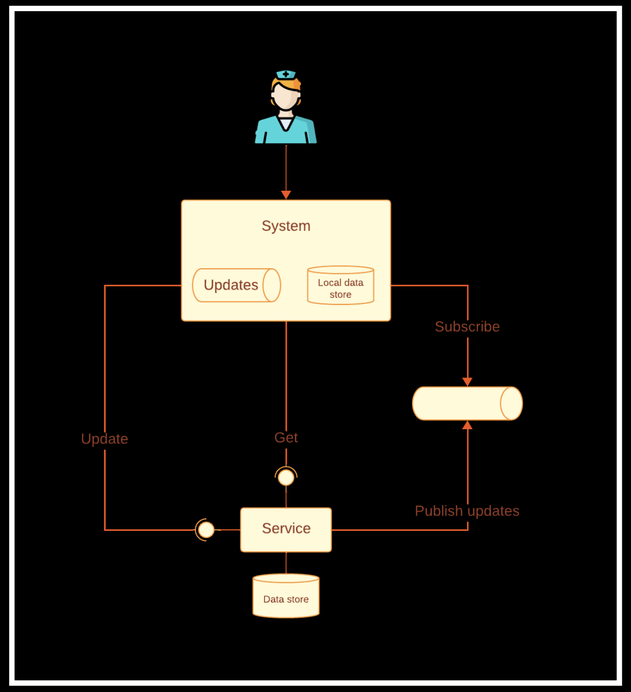
CADEC 2024.01.18 & 2024.01.24 | CALLISTAENTERPRISE.SE

CALLISTA

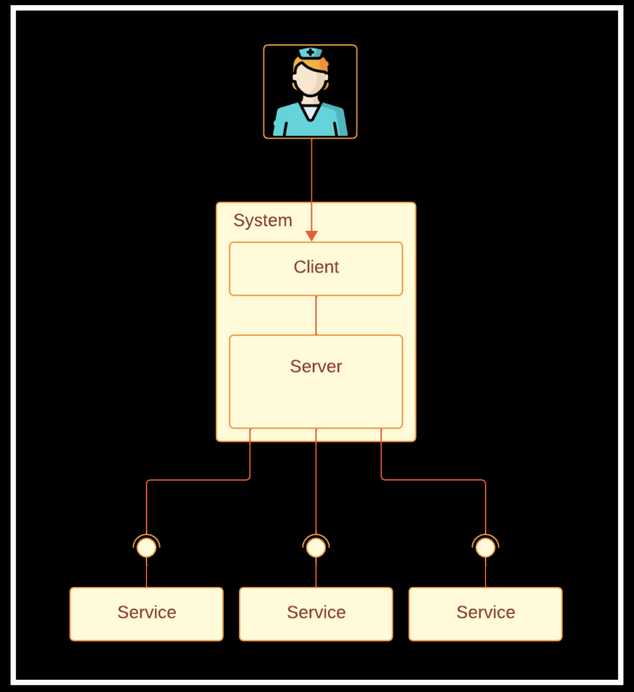
BASED ON A TRUE STORY...



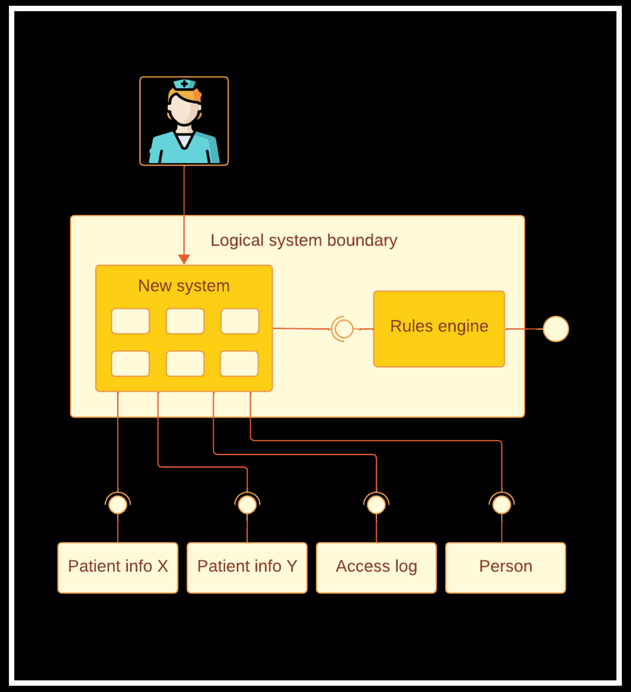
ON THE AGENDA:



1: THE HIGH-AVAILABILITY ARCHITECTURE

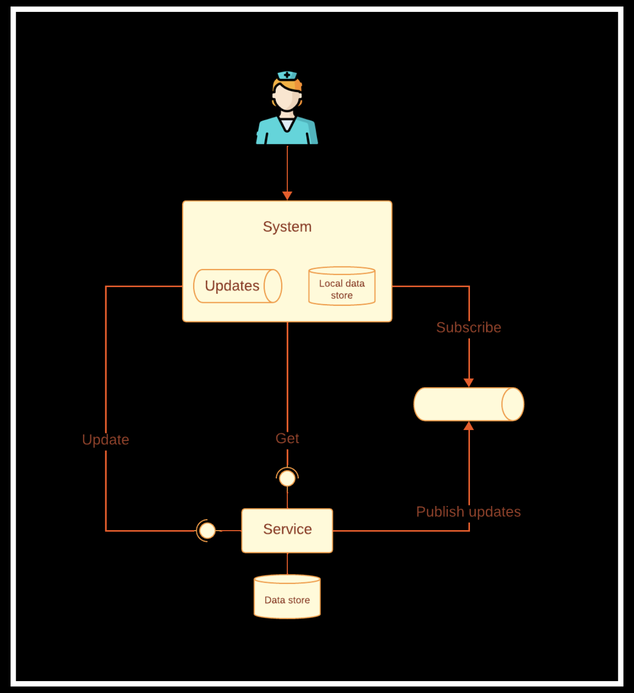


2: THE EXISTING SYSTEM



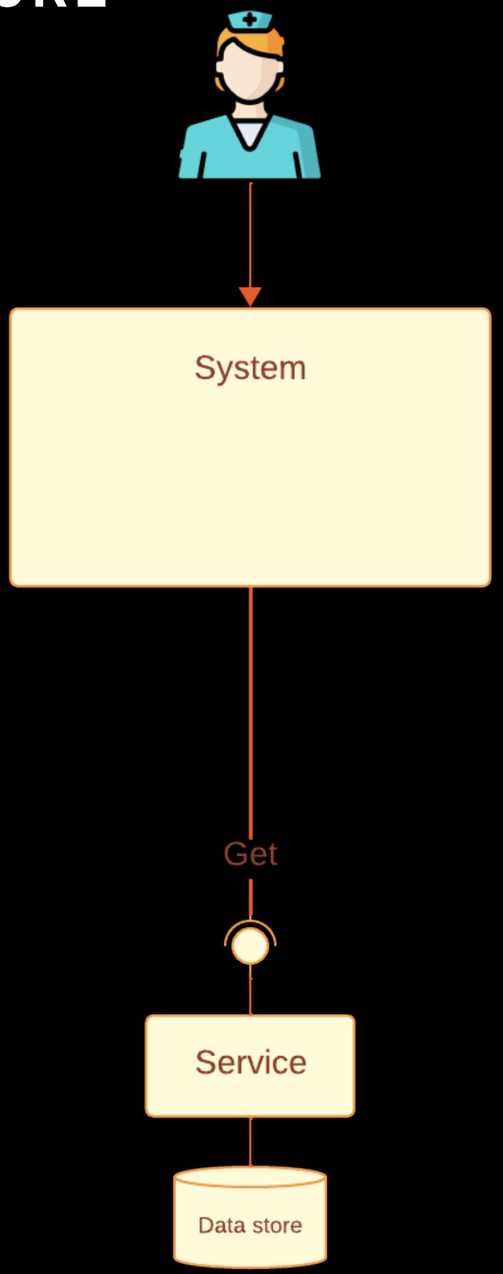
3: THE NEW SYSTEM

THE HIGH-AVAILABILITY ARCHITECTURE



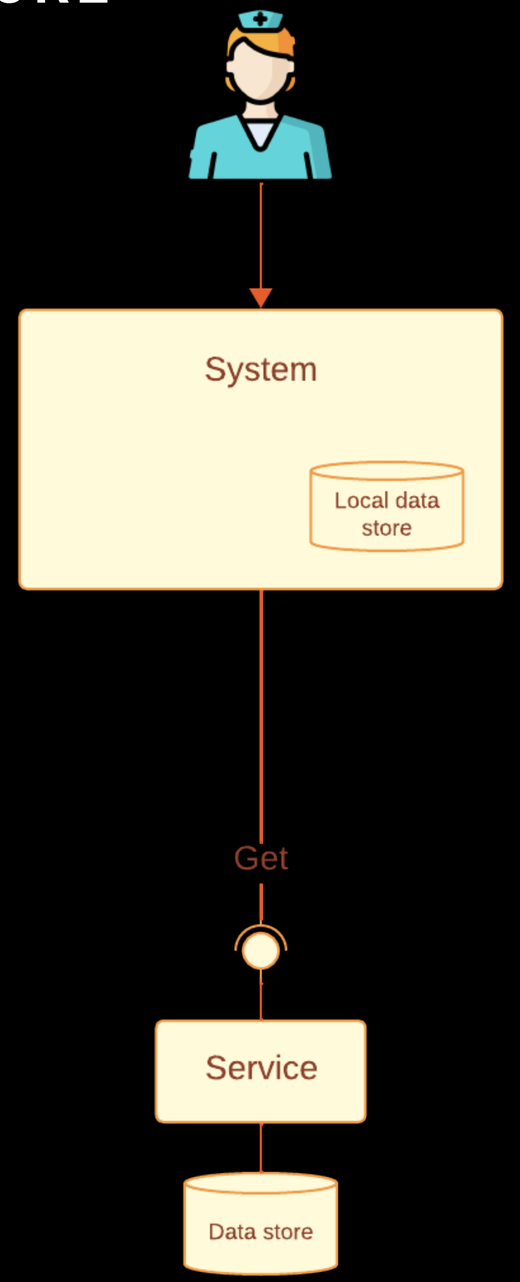
HIGH AVAILABILITY IN A DISTRIBUTED ARCHITECTURE

- Increase *time autonomy*



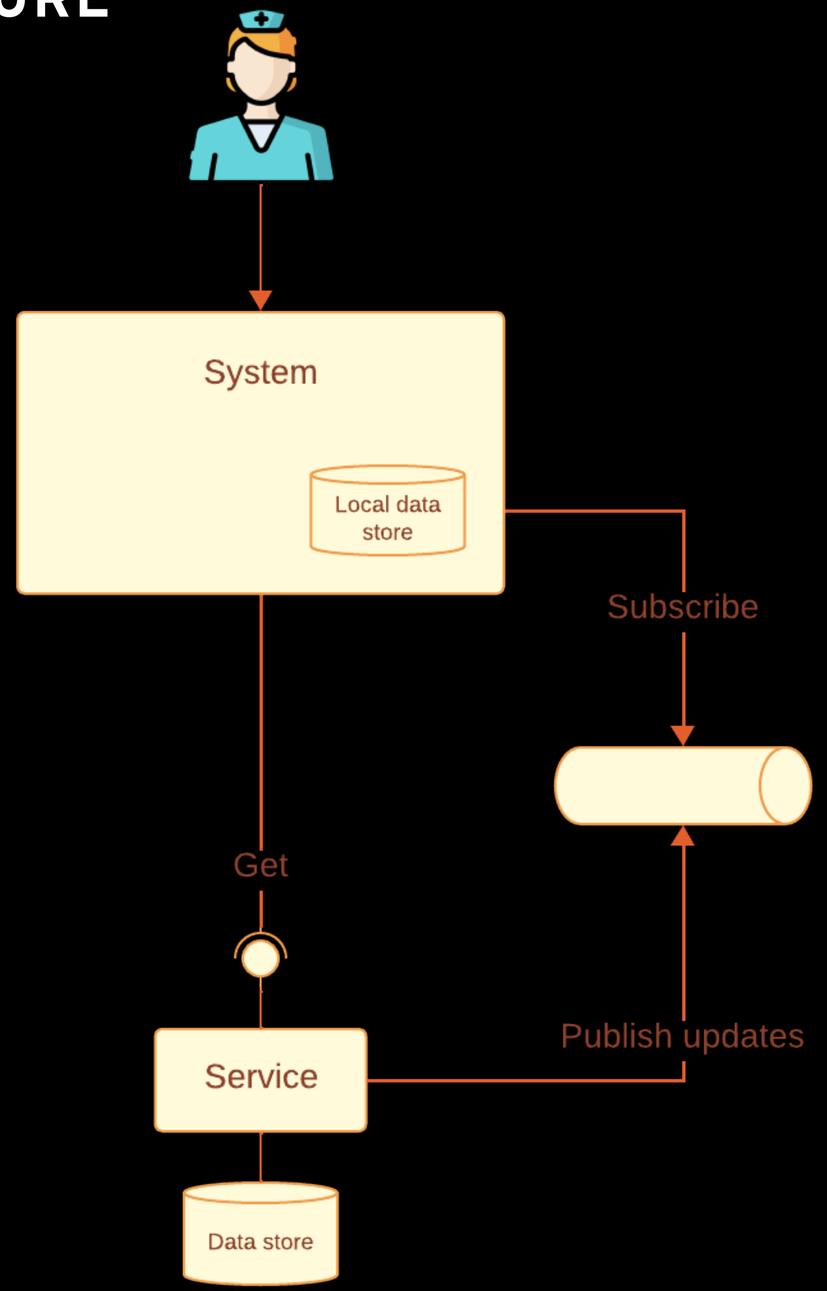
HIGH AVAILABILITY IN A DISTRIBUTED ARCHITECTURE

- Increase *time autonomy*
  - Keep data close (cache, read-only)



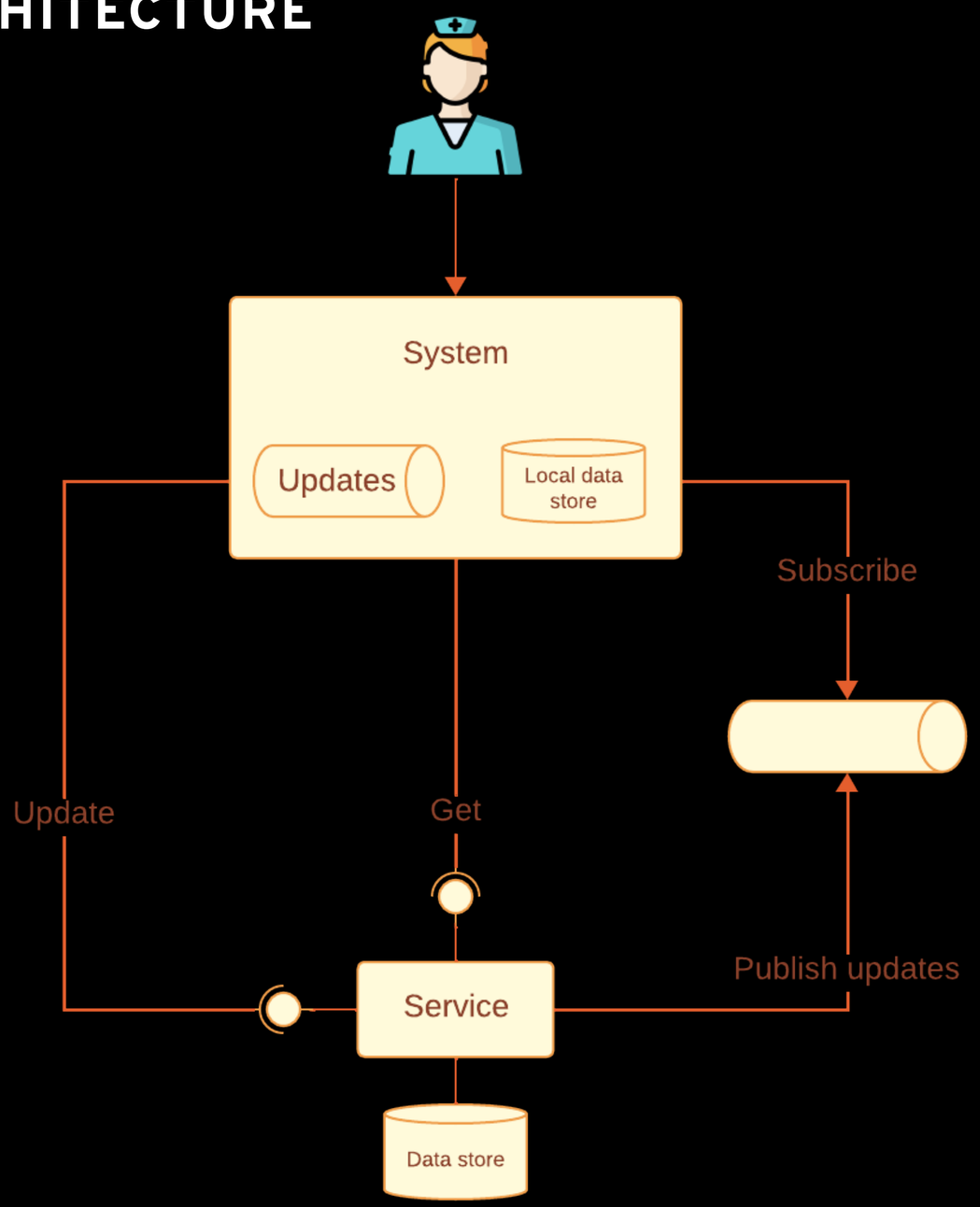
HIGH AVAILABILITY IN A DISTRIBUTED ARCHITECTURE

- Increase *time autonomy*
  - Keep data close (cache, read-only)
  - Pub/sub for receiving updates from master



HIGH AVAILABILITY IN A DISTRIBUTED ARCHITECTURE

- Increase *time autonomy*
  - Keep data close (cache, read-only)
  - Pub/sub for receiving updates from master
  - Asynchronous flow for updates to master



HIGH AVAILABILITY IN A DISTRIBUTED ARCHITECTURE

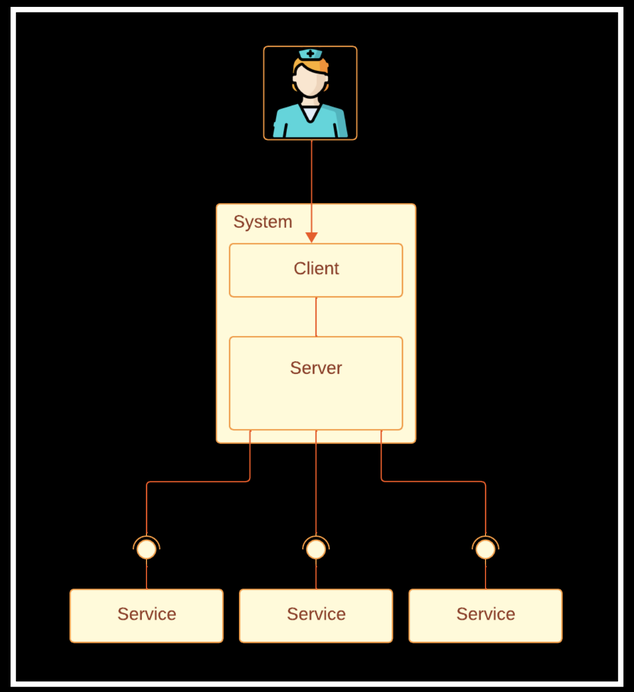
- For services that provide functionality, accept tighter coupling (no time autonomy) or duplicate logic

- Use resilience mechanisms
  - Time limiter

  - Retry

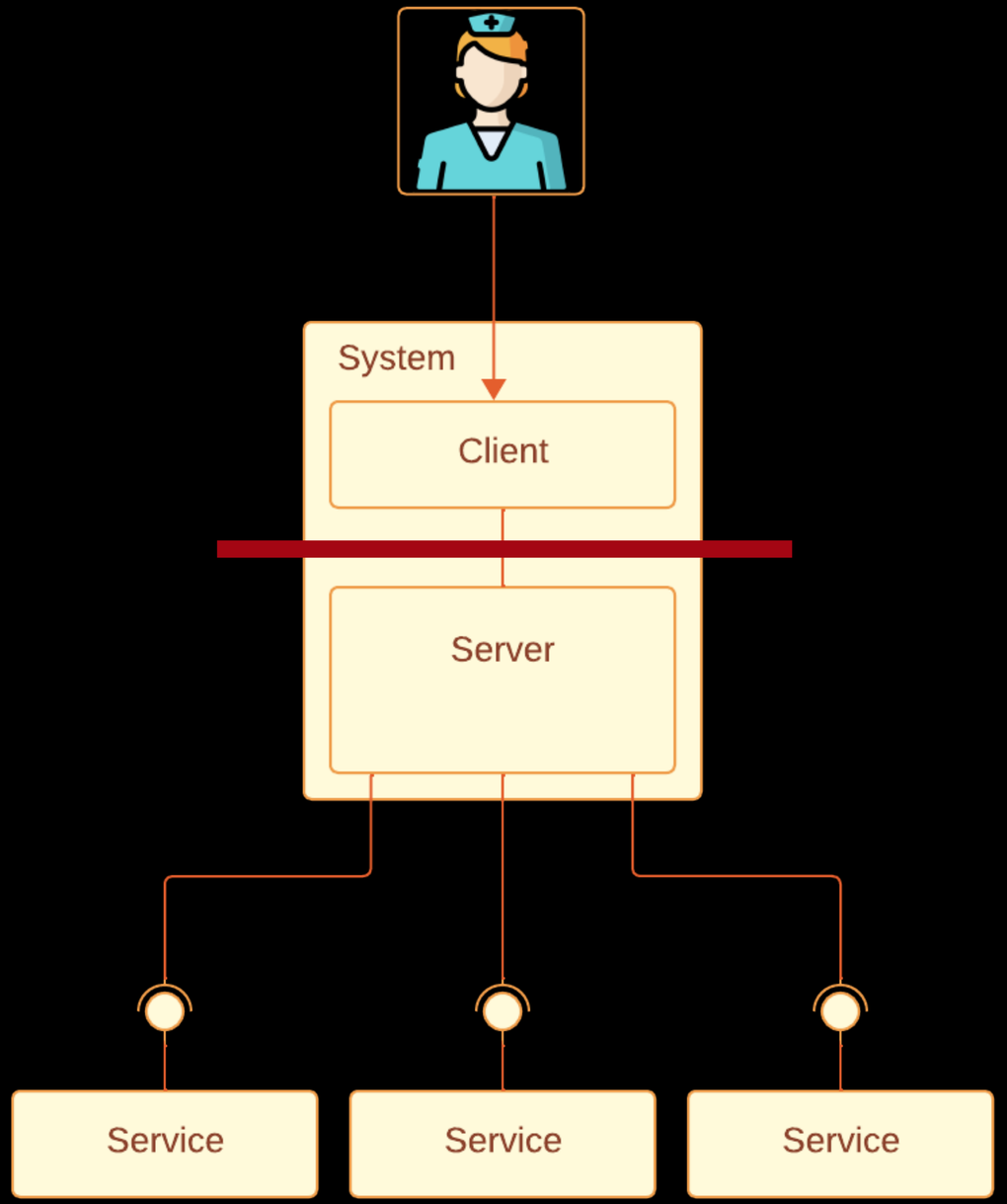
  - Circuit breaker

THE EXISTING SYSTEM



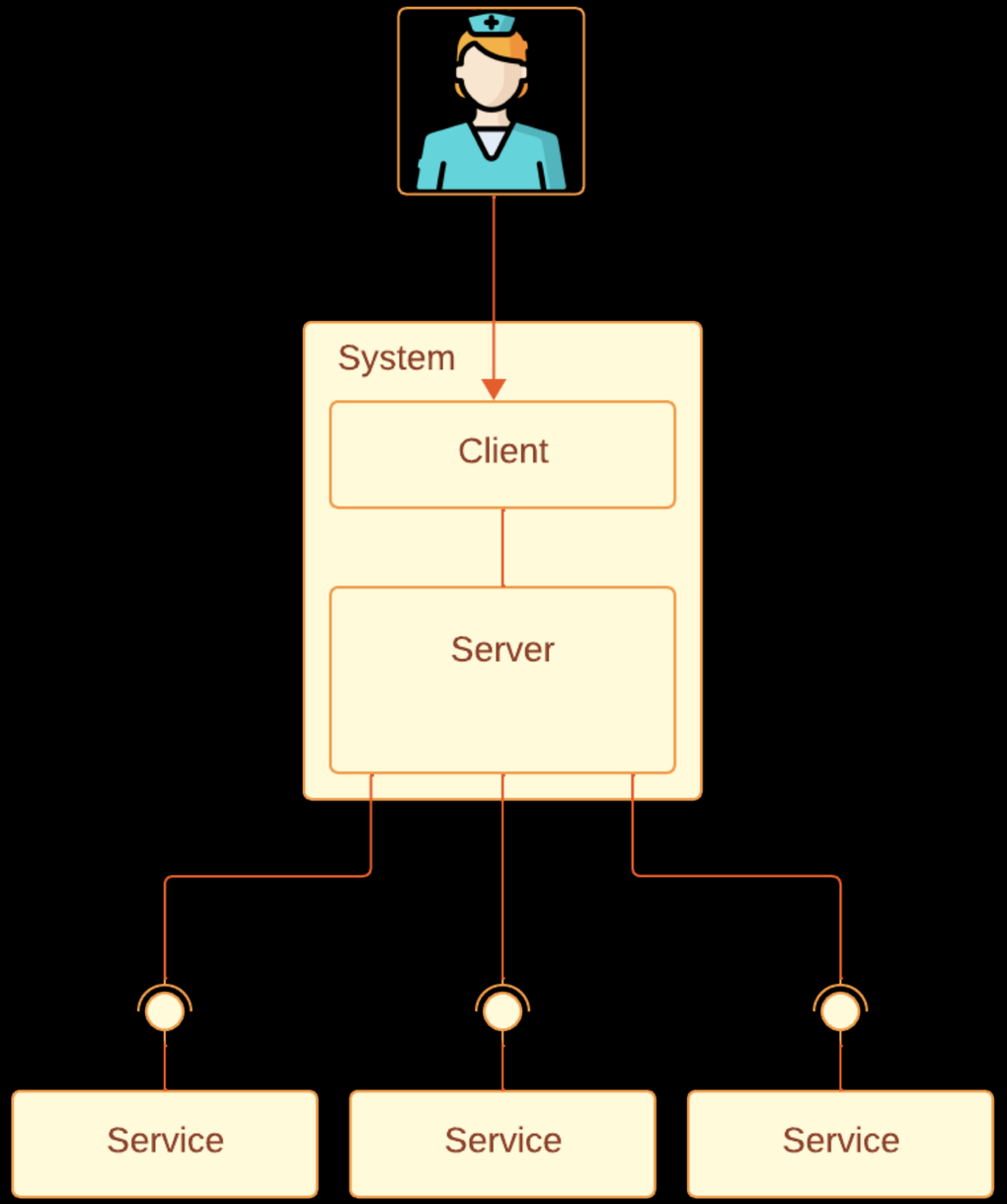
THE EXISTING SYSTEM

- Client-server
- Dependent on a couple of services (no resilience mechanisms used)
- Offline mode (client cuts connection to server)
  - limited functionality
  - synch problems

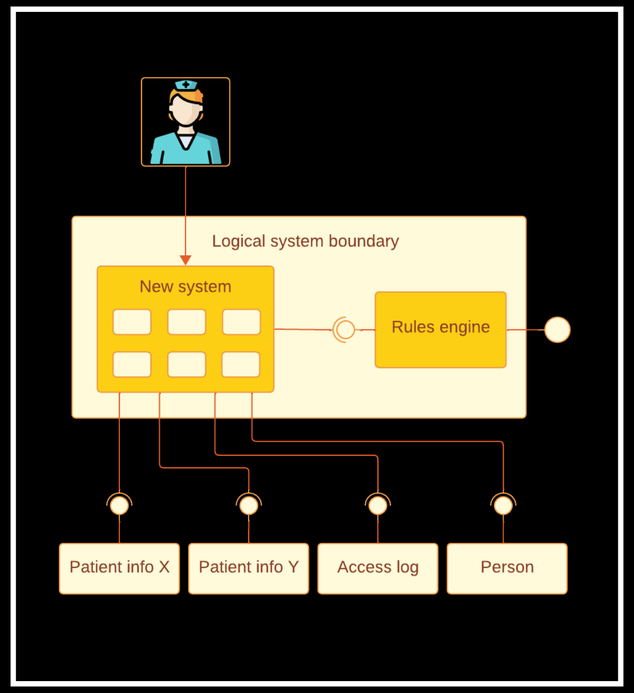


THE OLD SYSTEM - MAIN PROBLEMS

- Old and hard to maintain
- In need of some further development



THE NEW SYSTEM

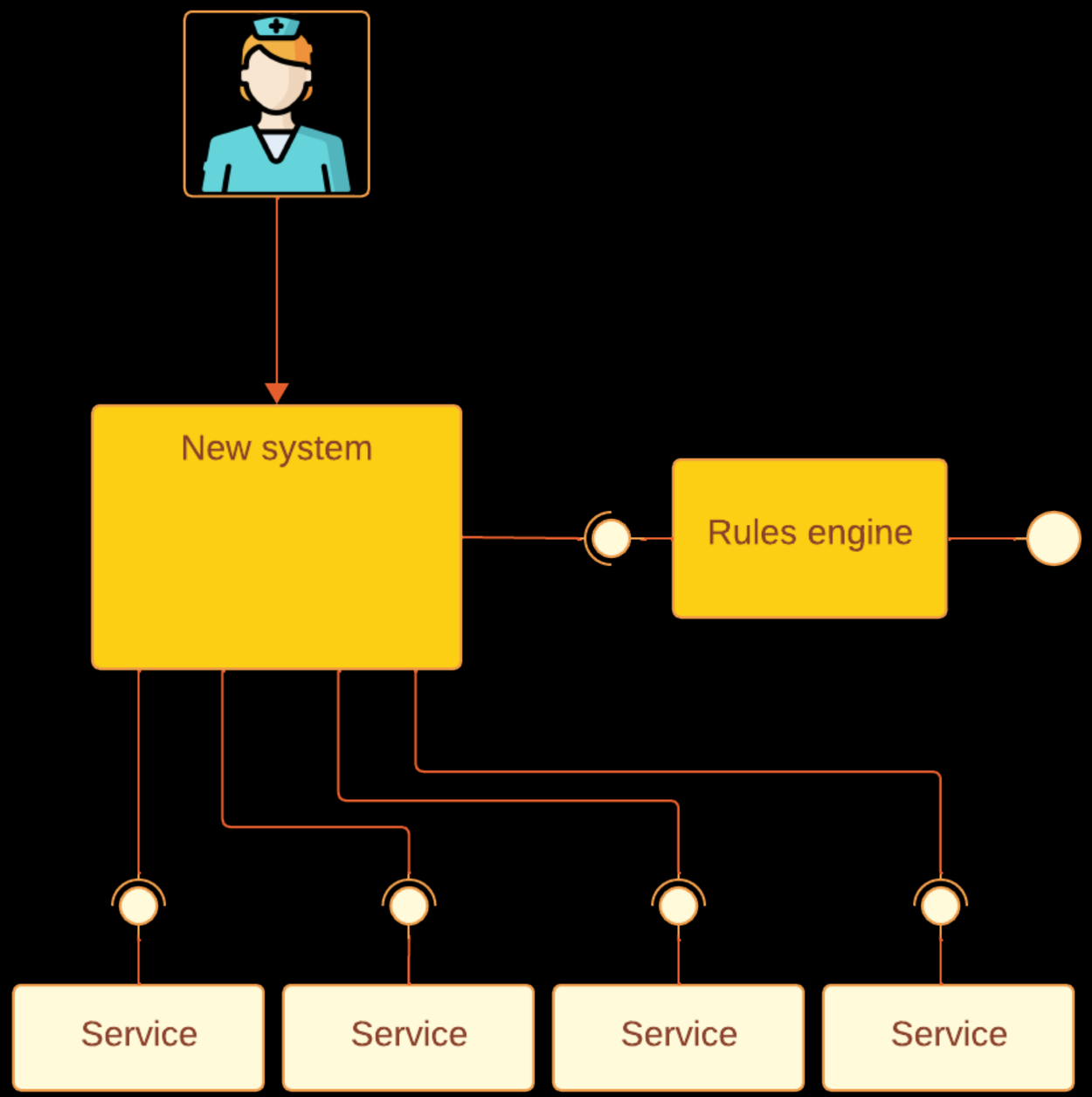


GOALS FOR THE NEW SYSTEM

- Maintain a high availability
- Integrate with more of the available services
- Modularize & offer new services to other systems
- Simplify maintenance

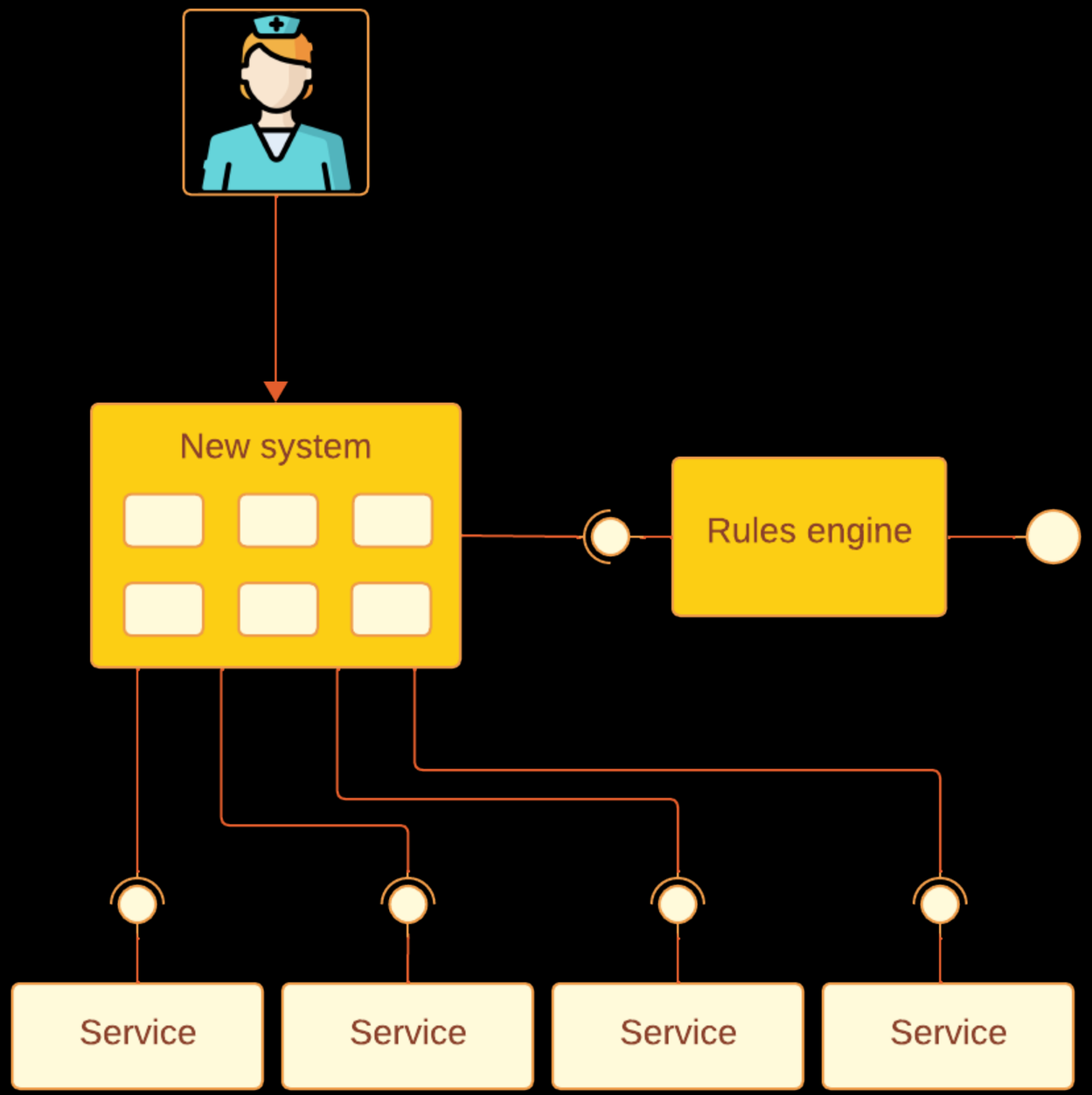
THE NEW SYSTEM

- New functionality, a rules engine, might be of interest to other systems
- A separate system to its other stakeholders
- A subsystem to us
- On what terms do we access the subsystem?
  - same as system-external services?
  - same as system-internal services?



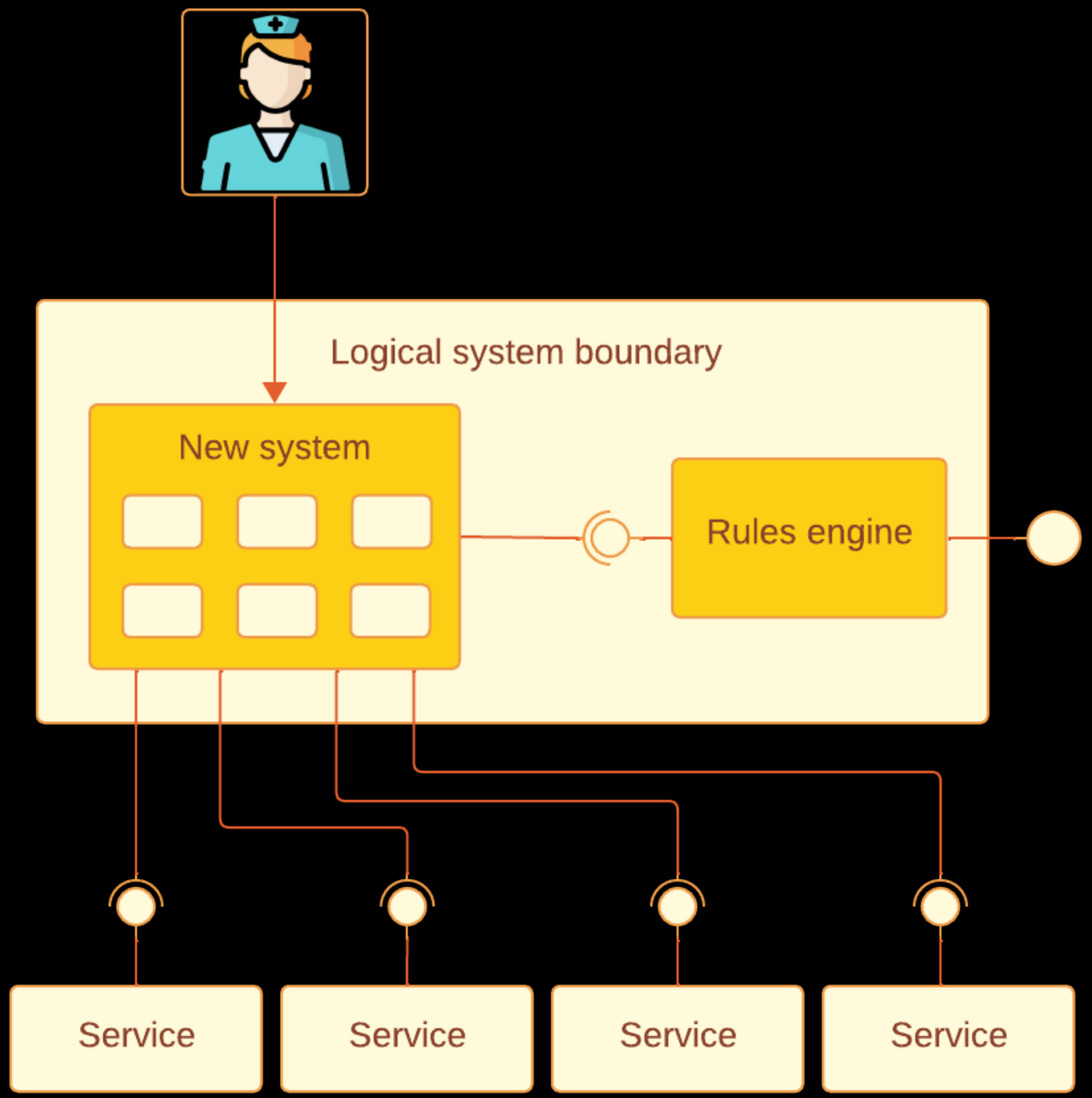
THE NEW SYSTEM

- New functionality, a rules engine, might be of interest to other systems
- A separate system to it's other stakeholders
- A subsystem to us
- On what terms do we access the subsystem?
  - same as system-external services?
  - same as system-internal services?



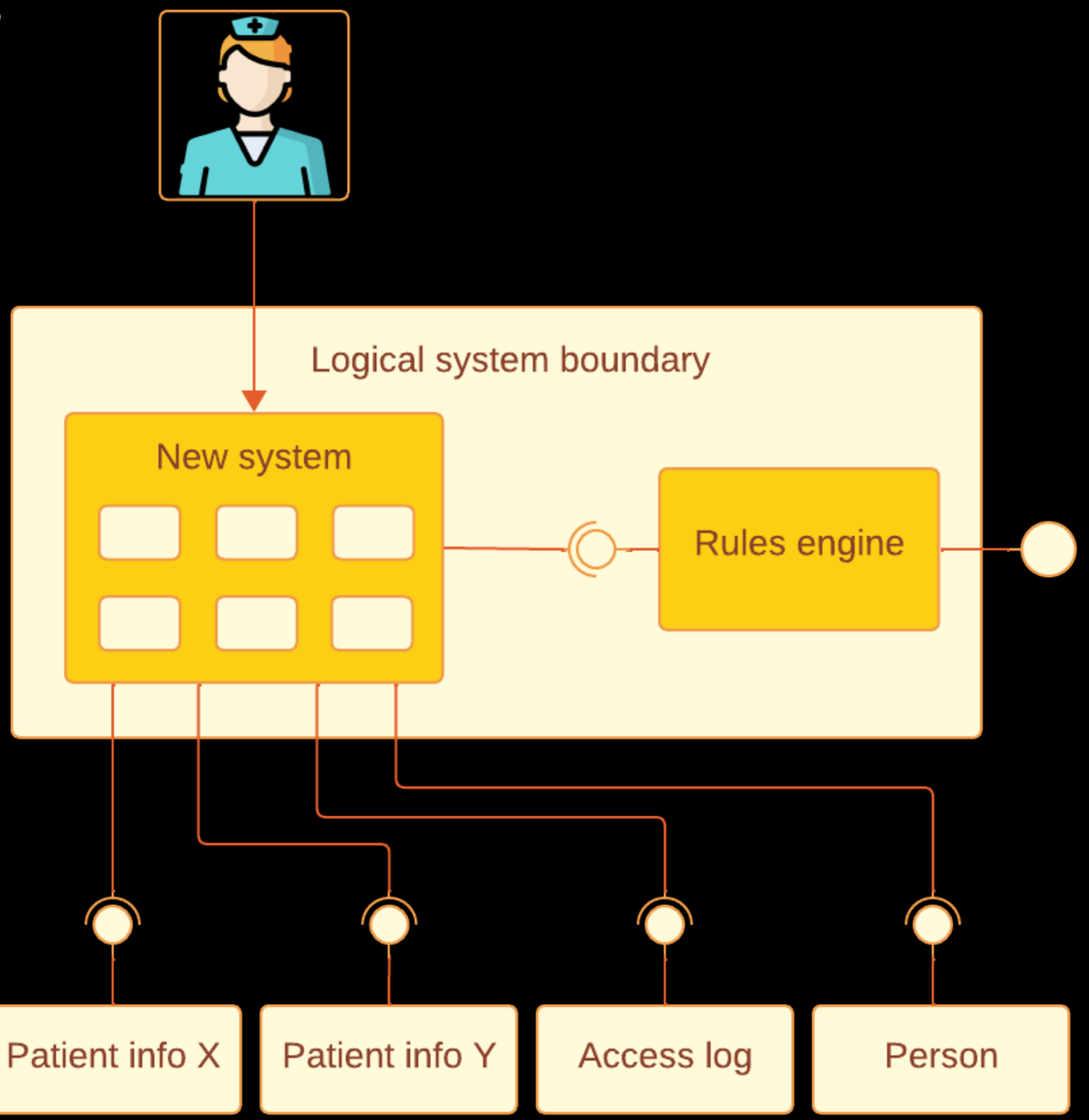
THE NEW SYSTEM

- Decision:
  - Subsystem can be accessed on the same conditions as components in the main system
  - A logical system boundary is defined, which gives us a context for defining conditions for the components inside the boundary



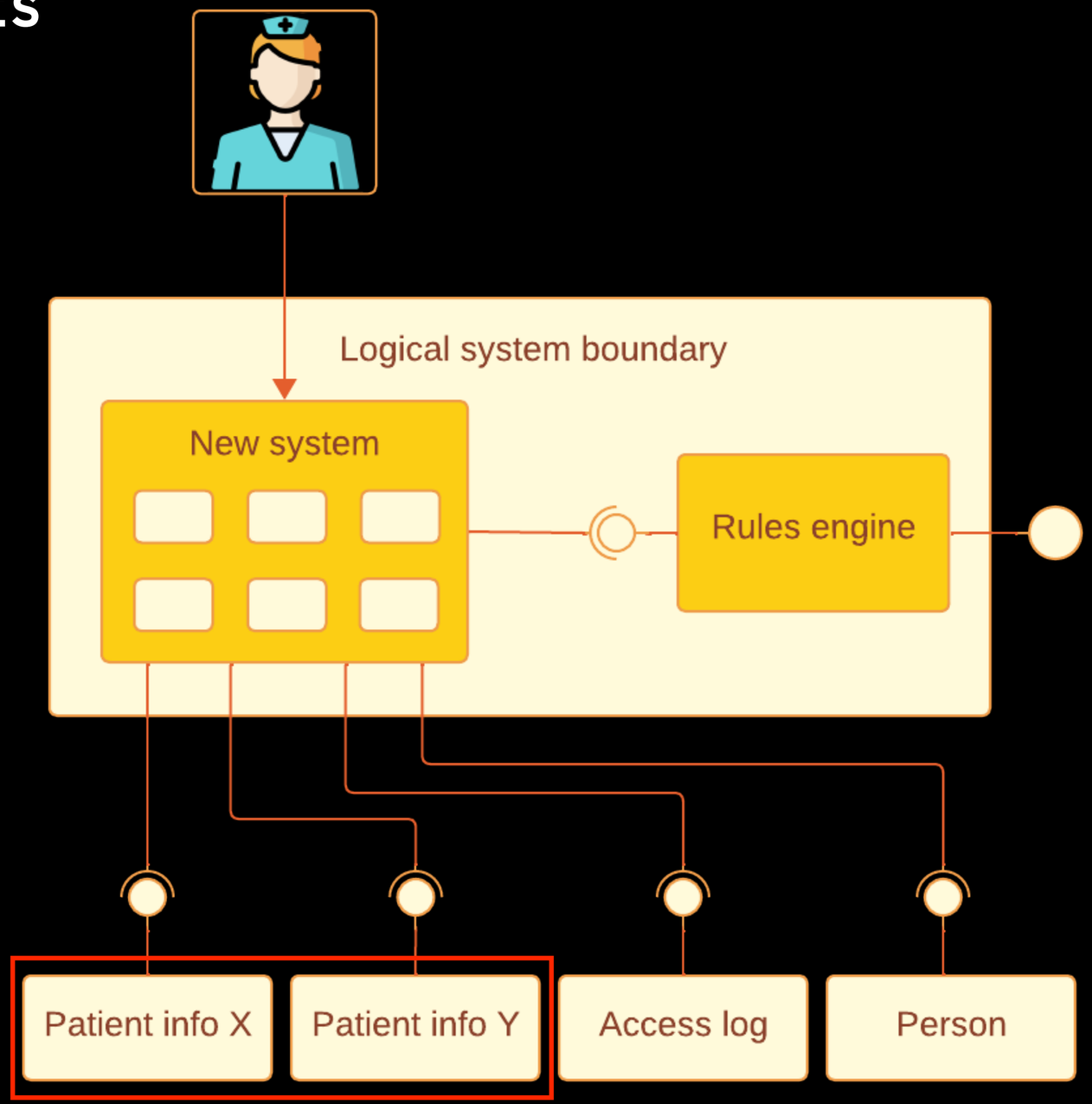
ANALYSING EXTERNAL DEPENDENCIES

- Three parts
  - Services that provides information about a patient
  - Access log, where we create data
  - Person service
- Existing services with multiple consumers
- Not that easy to adapt to new requirements



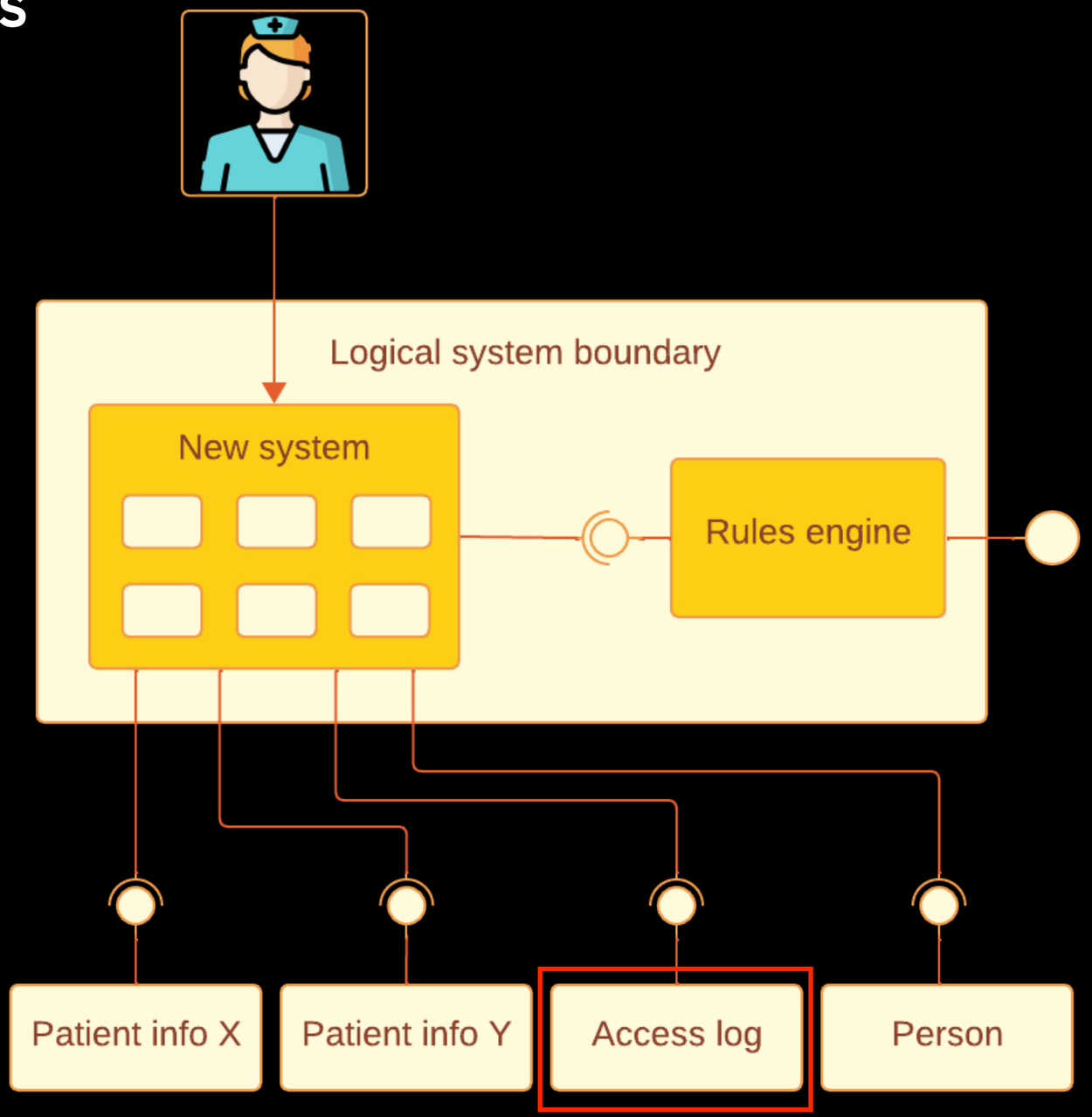
ANALYSING EXTERNAL DEPENDENCIES

- Services that provide information about a patient
  - getXForPatient(patientId)
  - Not mandatory (phew!)
- Decision: OK, but use resilience mechanisms



ANALYSING EXTERNAL DEPENDENCIES

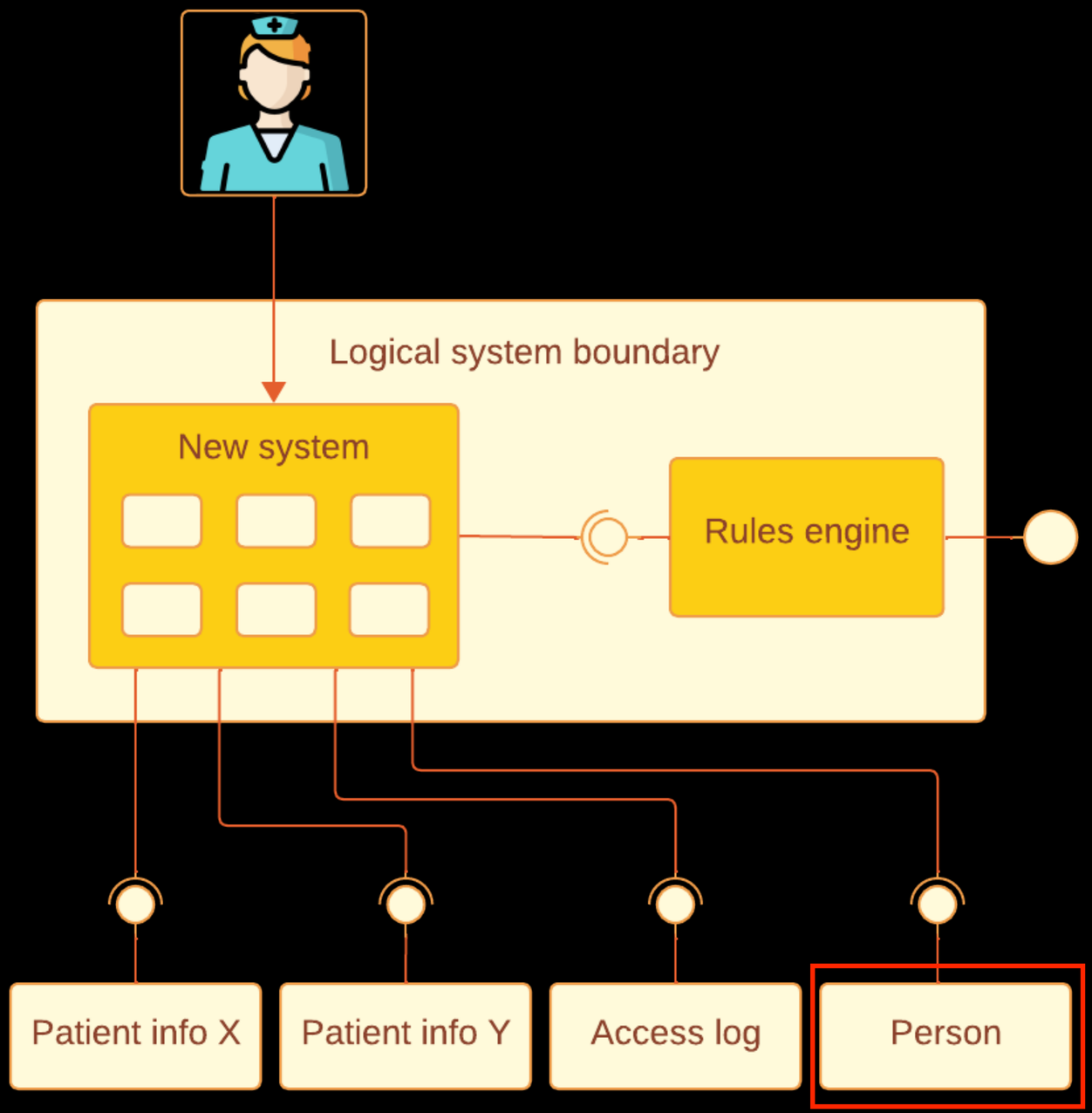
- Create: access log (who has seen what information about a patient)
  - ”small batch”
- Decision: asynchronous flow
  - low risk of error



PERSON SERVICE

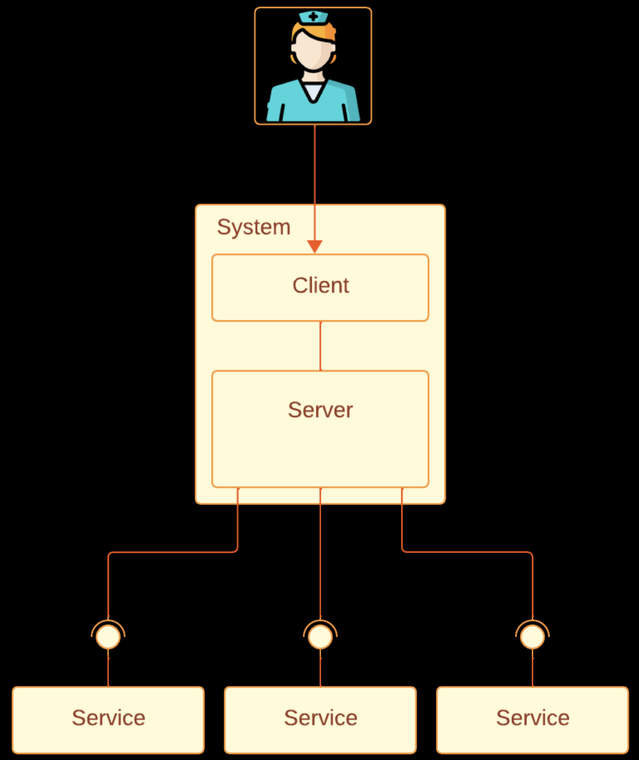
- Person information from Skatteverket (Swedish tax agency)
- Contact information
- ”Reserve id”

- Mainly a data store, but also provides some functionality



PERSON SERVICE - EXISTING SOLUTION

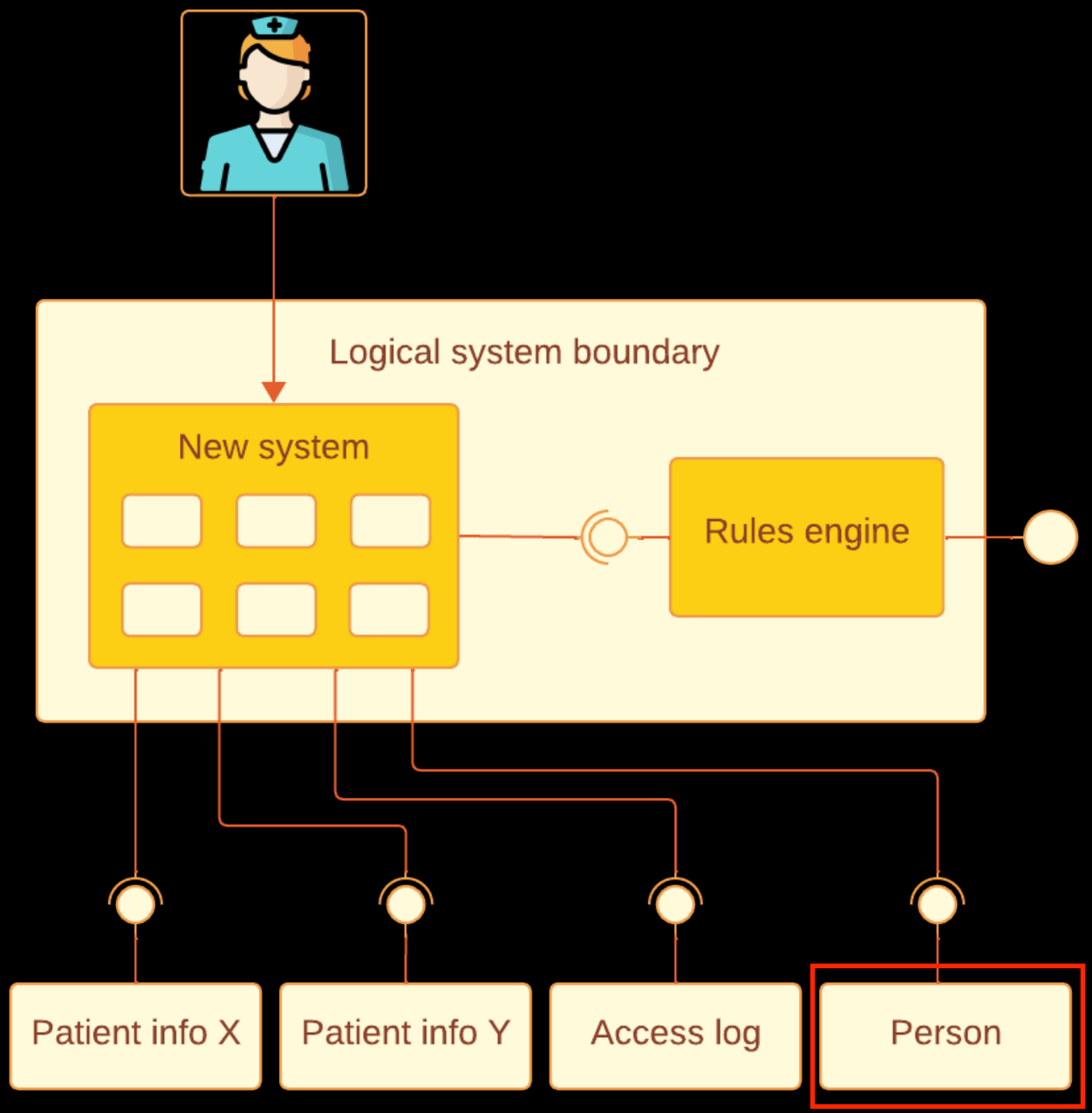
- Cache - but not available in offline mode
- Still a need to call person service in certain situations - no complete time autonomy
- Missing some functionality
- Solution originally designed for reading files directly from Skatteverket



PERSON SERVICE - NEW SOLUTION

- Duplicate data and functionality, or depend on service availability?

- Decision: Rely fully on person service
  - Person service is of highest availability class (no service windows)
  - Person service implements desired functional services



CLOSING WORDS

- Know the ideal architecture for realising your primary goals
  - and know the tradeoffs for that architecture
- Make decisions based on your specific circumstances
  - Organisational
  - Architectural

That's all folks!

CALLISTA