Data Loader Ideation.

Current Scenario:

* Each application encapsulates and solves a domain problem (iContract for contract management, iSupplier for supplier management etc.) with domain expertise built into it.
* Each application uses its own system of records to persist this domain information, tenant configuration, application configuration and data in a way best suited for it.
* Practically, an enterprise application cannot have a clear bounded context of domain (due to size of business features) and hence integrates with other application(s) to leverage their functionalities, providing client with aggregate features.

For example: iSupplier gets some it’s functionalities like central data management with integration to CMD

* Business evolution has resulted in applications coordinating in a point-to-point fashion with few mixed models like pub-sub or data replication depending on their requirements.

For example: eProc uses JMS for integration with certain products.

iSupplier replicates supplier data in solr and mongo for enabling other application to search and obtain suppliers. Products also use direct API calls.

* Models of domain and their operations are deeply embedded within each application making it difficult for services to manipulate these domain objects without being a part of it.

For example: A complete supplier representation from iSupplier has supplier details, supplier rules, supplier views/sub-views/attributes, reference data etc.

* Some high-level external APIs are exposed enabling bulk uploads, external integrations, service requests/calls etc.

Problems Faced:

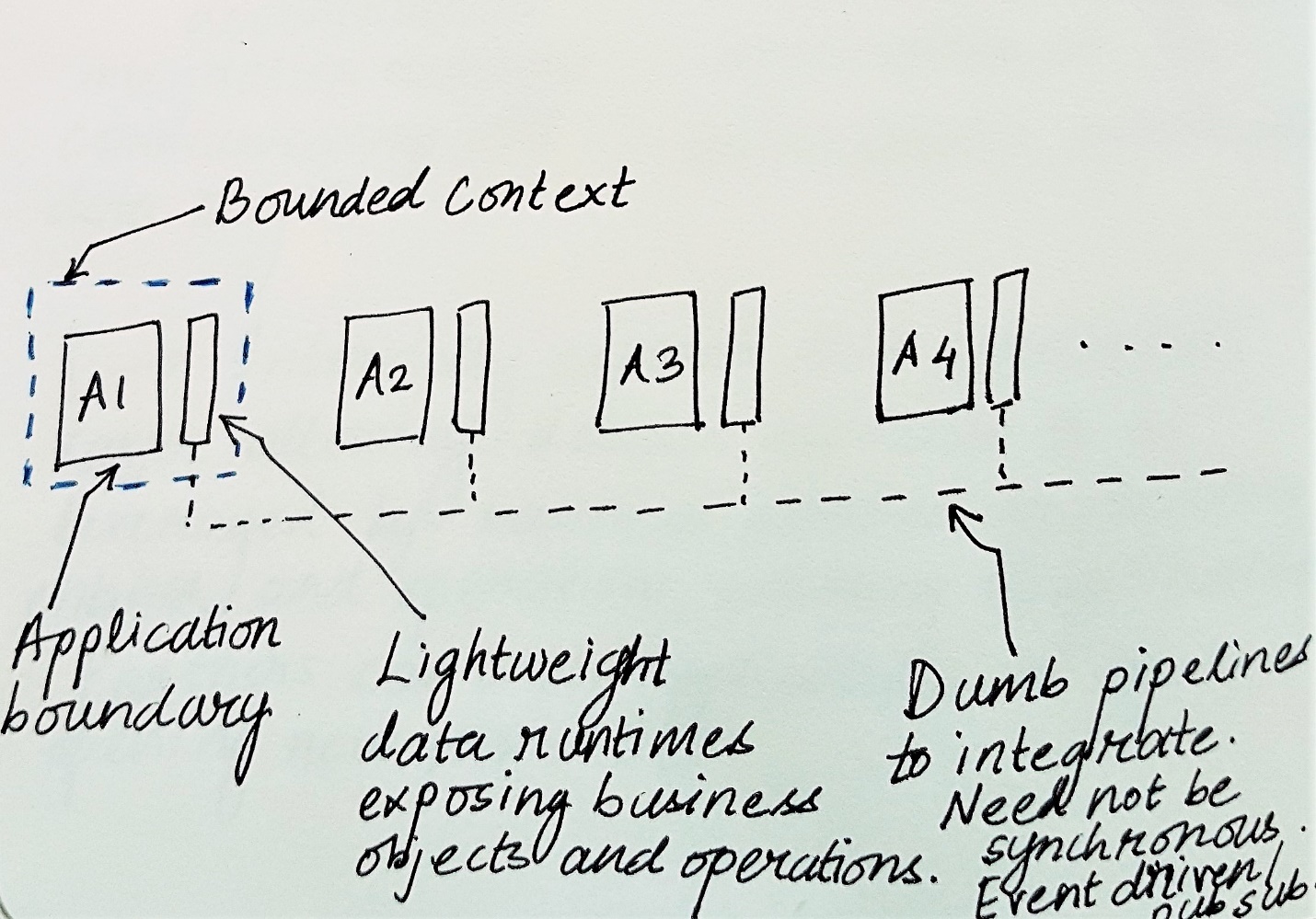
* Tenant migration and data migration between setups and environments is difficult, error ridden and time consuming. Analysts spend a lot of time solving these problems.
* Cannot migrate a tenant from one environment to another across complete Zycus suite in one click.
* Slow creation of completely configured QA setups and test bed creation.
* Scenario based automations that are slow to develop ( because of time needed to understand product’s system of records ), fragile, not standardized and non-reusable. The time spent on product’s KT is not reused (requirements change, people change) resulting in huge wastage of time.
* Involvement of multiple teams (Development team, L2, DBA, RM) with a slow process in executing CSCRs.
* No tenant configuration/data versioning (Environment x Product x Tenant) and pipeline line like promotion for deploying usable products with client data already configured.
* Missing the ability to stage a product’s replica configured with production data for analyzing/debugging an issue on production and hot fixing.
* Partner projects like RPA, integration, API gateway, infrastructure abstraction, analytics are slow and cannot be realized to full potential because there is no standard way of communicating with apps and their data models.

Proposal:

* Developing a distributed application for data-pipelining that is decentralized using light weight data runtimes.
* Each of these runtimes are developed for an application domain (one for iSupplier, one for iContract etc.) making the underlying data-structures and relationships easier to understand. These applications encapsulate data collection and translation, supporting batch and event architecture.

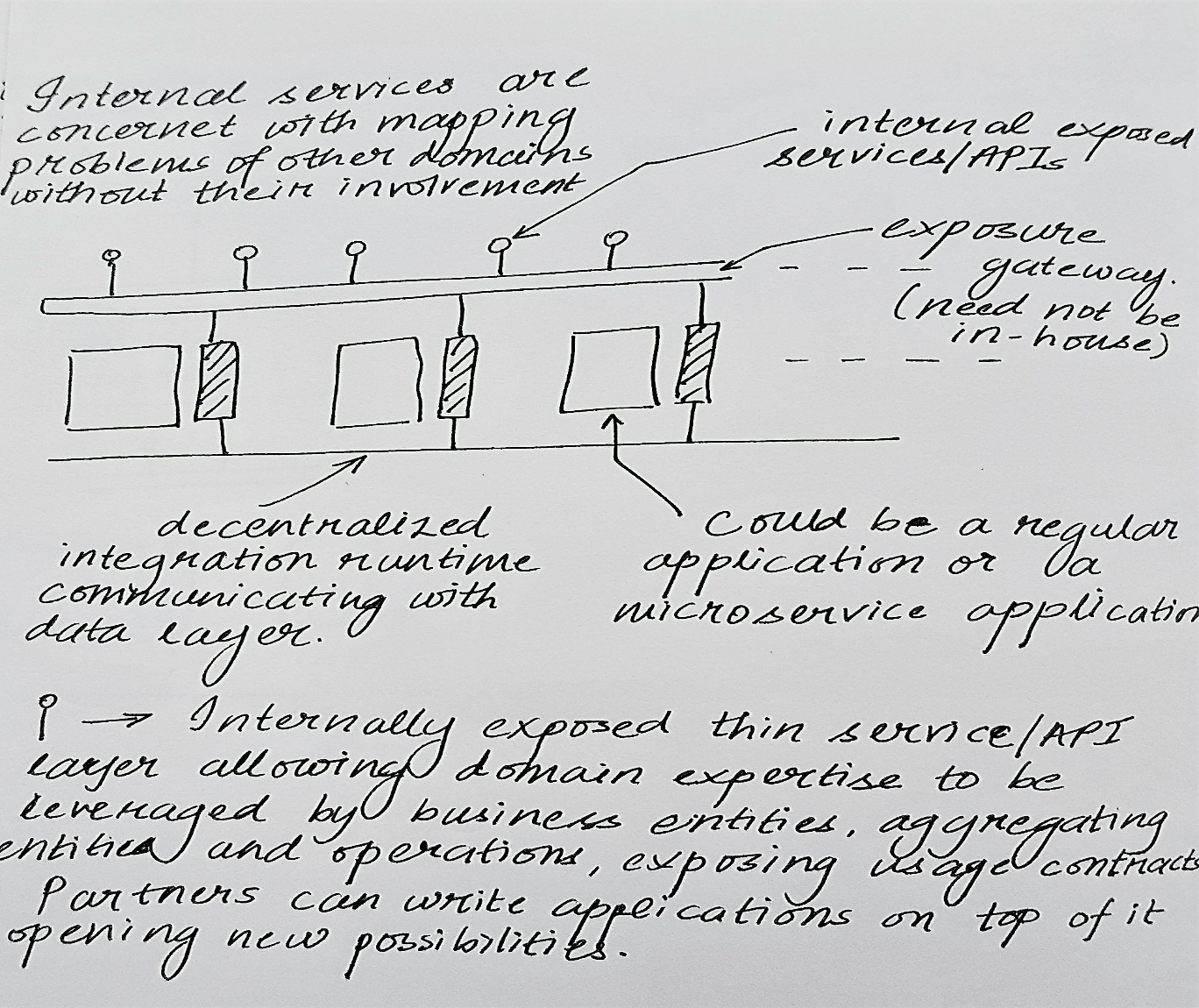
The underlying data-model is simplified down into a Logical Data Model (LDM) and an API is exposed to provide access and operations on LDMs.

This layer exposes low-level abstractions.

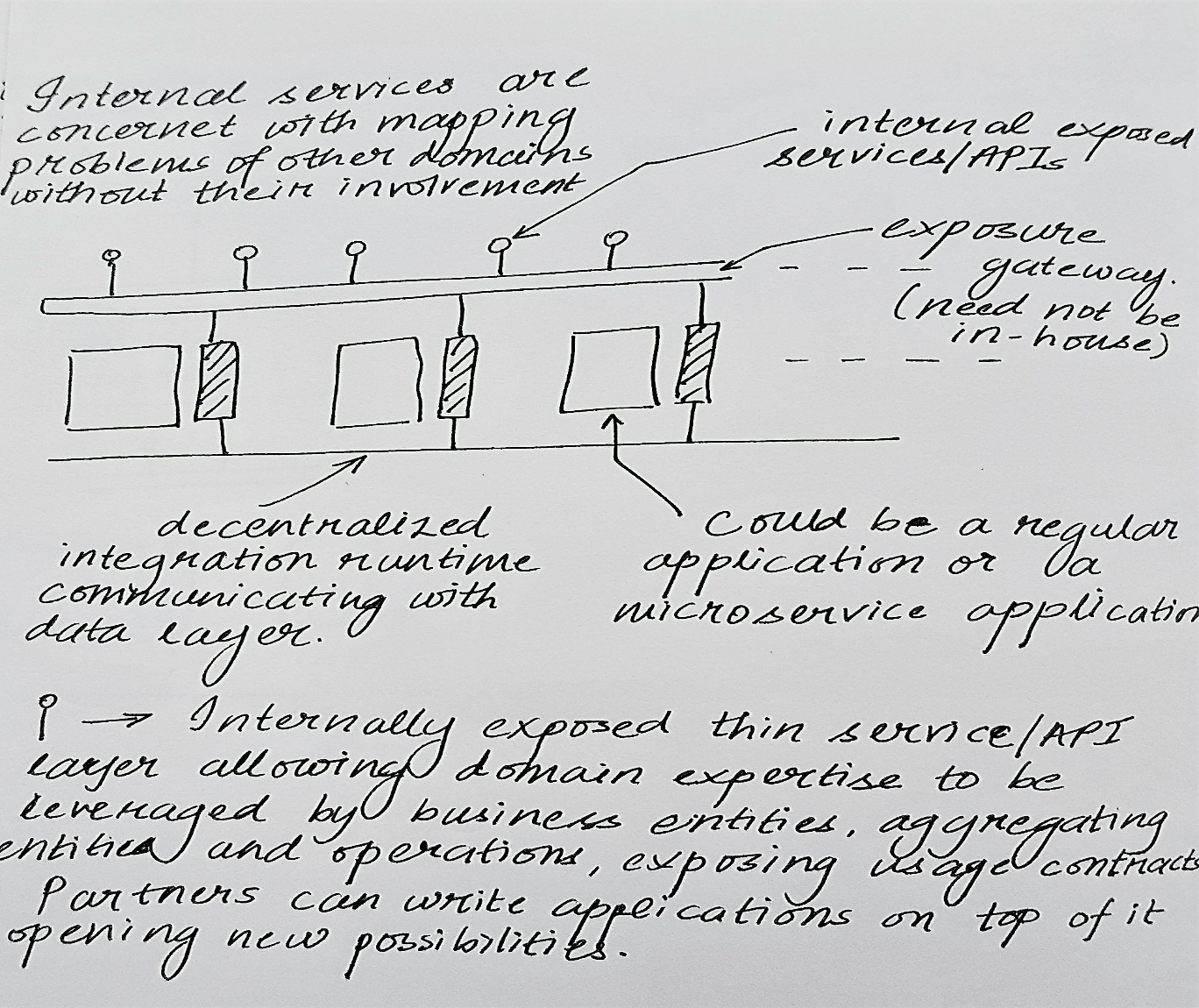


* Building a data-pipeline platform on top of data-runtimes, aggregating the low-level abstractions across business applications and enabling teams to build pipelines quickly without needing to know implementation details of applications involved. The communication with lower layers is done by compact serialization formats.

DSL for writing pipelines, workflow management, events framework, message bus, data versioning, query and visualization, monitoring and telemetry etc. will be built on top of this. This layer exposes high-level abstractions.



* Building logically centralized layer exposing Zycus Business Objects (ZBO), other domain models and operations, enabling the development of a palette of data adapters for Zycus products and integrating with external systems with simple drag-drop/configuration.
* Developing enterprise grade RPA framework which will enable quick development of new automations and reusability of existing ones.
* Loosely coupled API gateways - BFF style, supporting different capabilities and granularities of operations for different front-ends (web, mobile, programmatic clients), throughput throttling, authentication, encryption can be easily implemented or developed.
* Data-runtimes act as smart end points and coordination can be done with dumb pipelines, using any suitable design.



* The separate data pipeline application proposal is made considering below factors:
  + Separation of concerns and computational expenses.
  + Isolation of dependencies and errors.
  + Parallel development and non-interfering, easily coordinated releases. No point to point coordination between product teams.
  + Different consumers of data can have their needs mapped without explicit product dependency. Teams can build their own products/frameworks atop it like RPA, tenant/data migration (dev-uat-prod), test-bed creation, CDv2 – deploying usable applications with tenant already configured, staging setup creation with tenants for production hot fixes etc.

