Design Patterns and decisions taken:

**Use cases and decisions:**

**Batch Processing:**

In one of the use case, there are lakhs of records processed daily but in the current design implementation record by record processing is implemented and lot of time is wasted in processing these records. As an alternative design Batch processing is suggested by me and after implementing this lot of time got saved. And use of Transaction as and when needed.

**ScatterGatherer Vs Choice Router:**

When multiple messages are processed through to route to different destinations use of scatter gatherer works comparatively better as Parallel processing is accomplished and if any one of the messages among them fails it can be handled with the use of filters and care can be taken to stop the failure of all the messages. Choice Router can’t be used for parallel processing.

**Publish Subscribe Mode:**

With the use of Publish subscribe model if one of the publisher publishes then all the subscribers can listen to the same message. In one of my previous projects Subscriber is designed in such a way that multiple publishers can use the same subscriber and it will handle all the scenarios of different end systems efficiently.

**Generic Exception Handling for Multiple End Systems:**

In one of the use cases different End systems should publish the error emails to Business on daily basis. In this scenario, instead of each end system sending the messages separately I have designed a generic exception handling mechanism where it will pull the error records of different end systems once in a day and send the email to business. This reduced the load to all other systems as generic exceptional handler is taking care of sending emails.

**Use of RAML 1.0 Vs 0.8**

New RAML provides many new features when compared to that of 0.8 through which we can properly organize the data.

**Selecting the appropriate processing strategy based on the need:**

* + - **Synchronous Flow Processing Strategy**
    - **Queued Asynchronous Processing Strategy**
    - **queued-thread-per-processor-processing-strategy**
    - **thread-per-processor-processing-strategy**

**Usage of In-memory database:**

In memory database can be used to temporarily store the records till the business functionalities are performed on the data. With this implementation, there is low latency and high throughput. Cache implementation on top of this will drastically reduce the number of hits to DB.