packt

1ST EDITION

# Salesforce Anti-Patterns

Learn How to Create Great Salesforce Architectures from the Common Mistakes People Make on the Platform

LARS MALMQVIST

# Chapter 1

## Figures

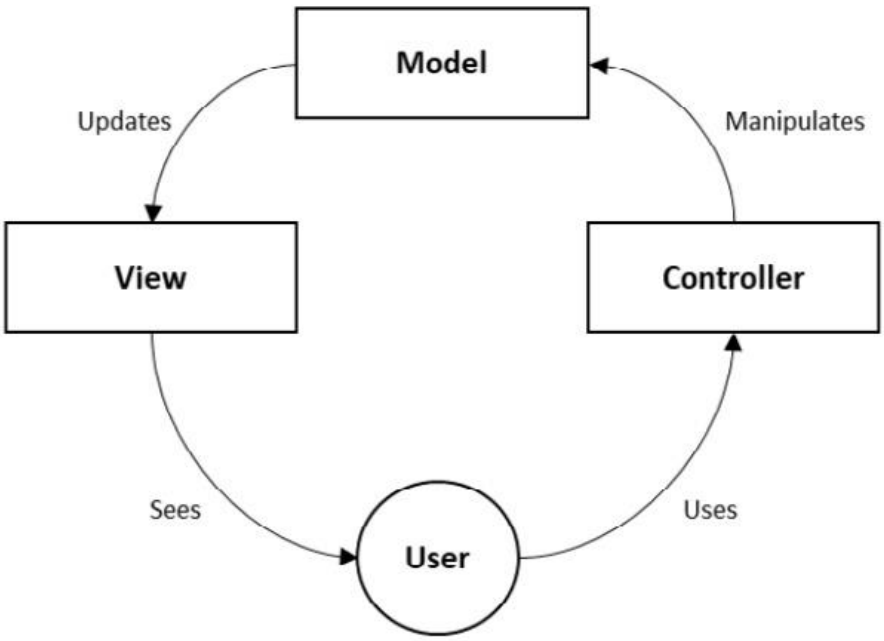


**Figure 1.1 – MVC pattern diagram**

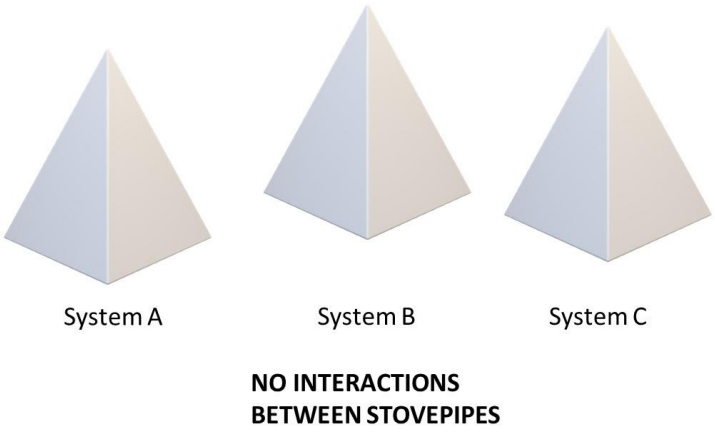


**Figure 1.2 – The Stovepipe anti-pattern**


**Figure 1.3 – The dangerous feeling one might have when engaging in the Hero anti-pattern**

# Links

The integration patterns guide lists all the main patterns to use when designing Salesforce integrations: https://developer.salesforce.com/docs/atlas.en-us.integration_patterns_and_practices.meta/integration_patterns_and_practices/integ_pat_intro_overview.htm.

The Salesforce Architects site, while new, contains a range of patterns across domains, from code-level specifics to reference architectures and solution kits to good patterns for selecting governance: https://architect.salesforce.com/design/#design-patterns

# Chapter 2

## Figures

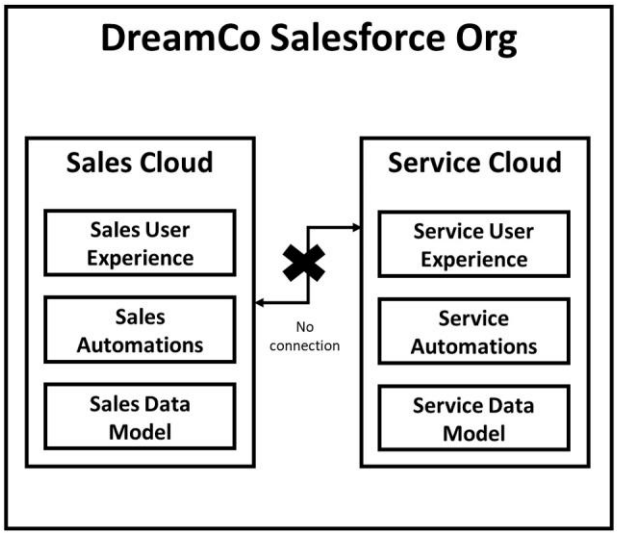


**Figure 2.1 – DreamCo's org after the initial implementations**

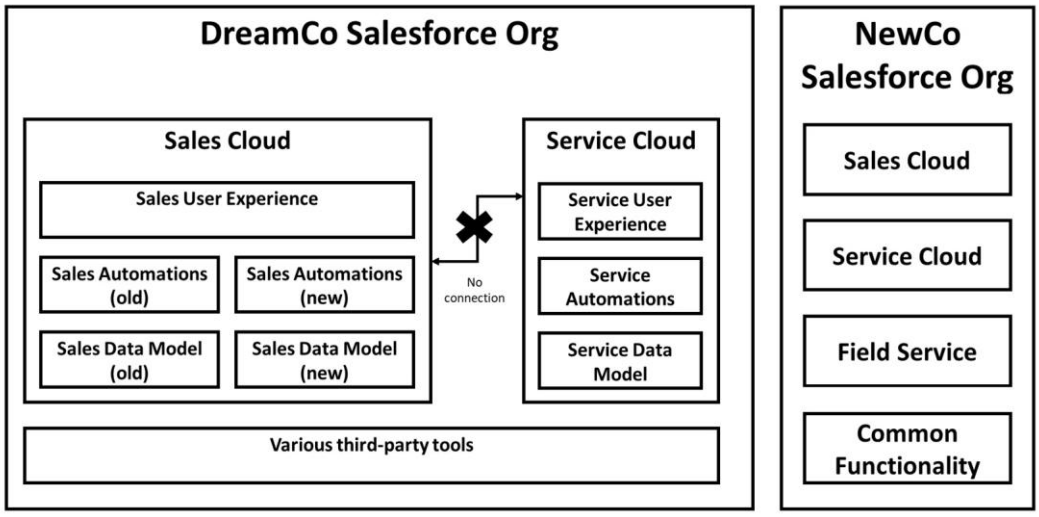


**Figure 2.2 – The DreamCo consolidation scenario**

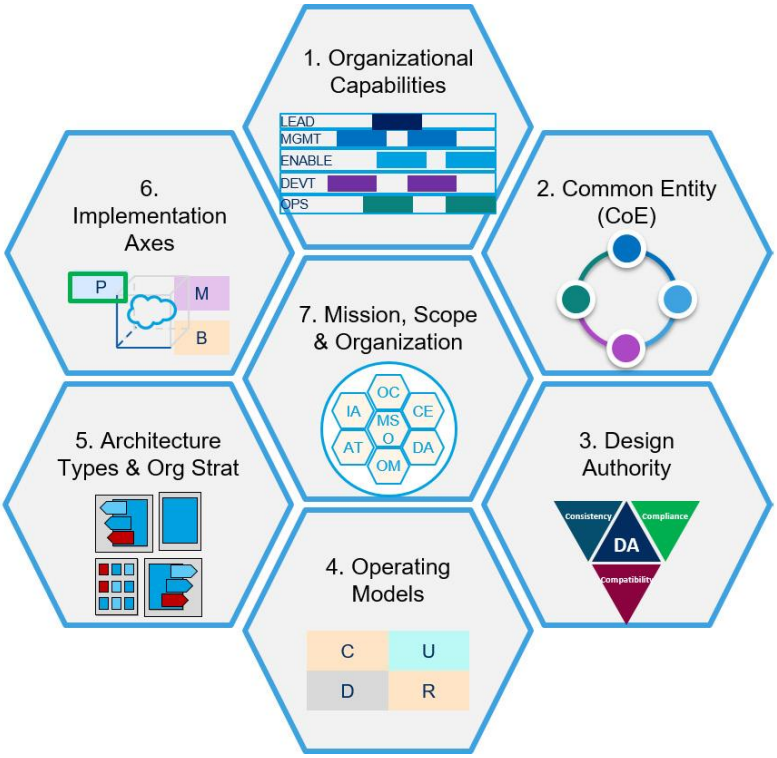


**Figure 2.3 – SOGAF framework elements**

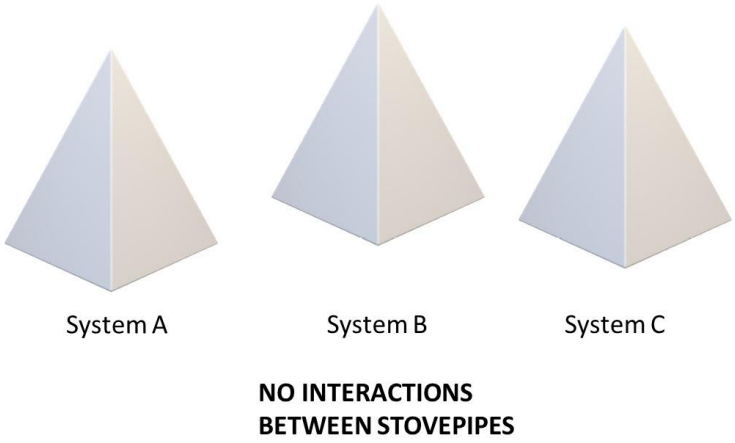


**Figure 2.4 – A landscape of stovepipes making up a stovepipe enterprise**

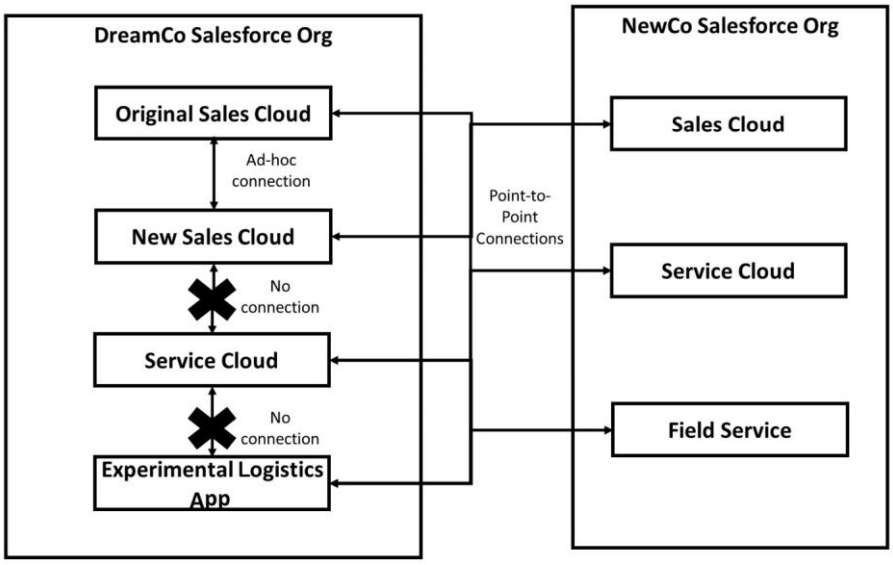


**Figure 2.5 – The DreamCo Big Ball of Mud architecture**

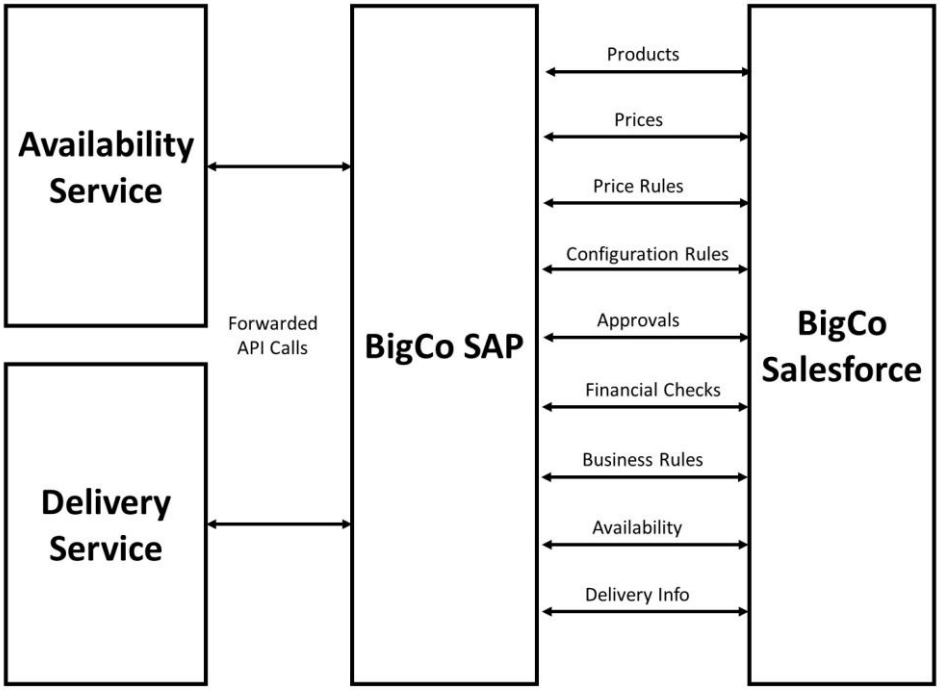


**Figure 2.6 – The BigCo integration scenario**

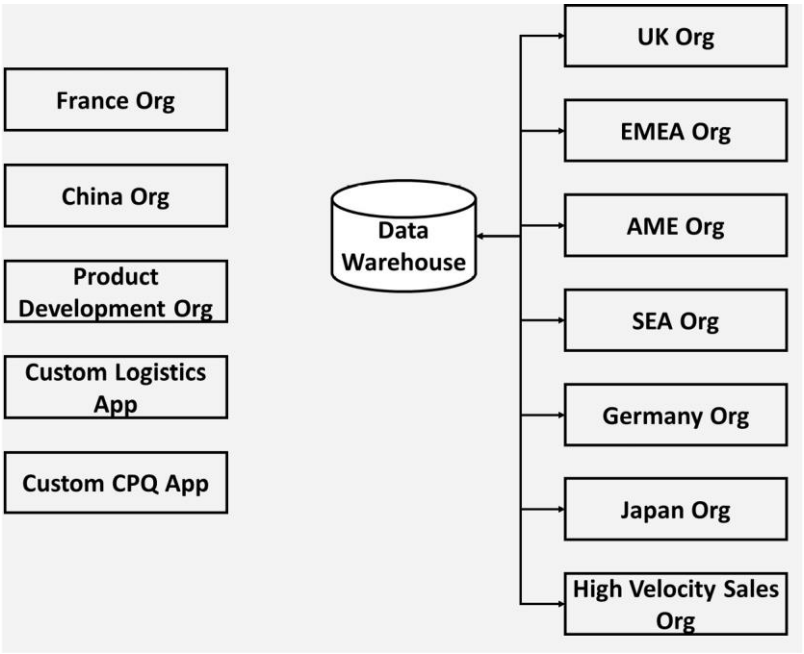


Figure 2.7 – The BigCo org overview

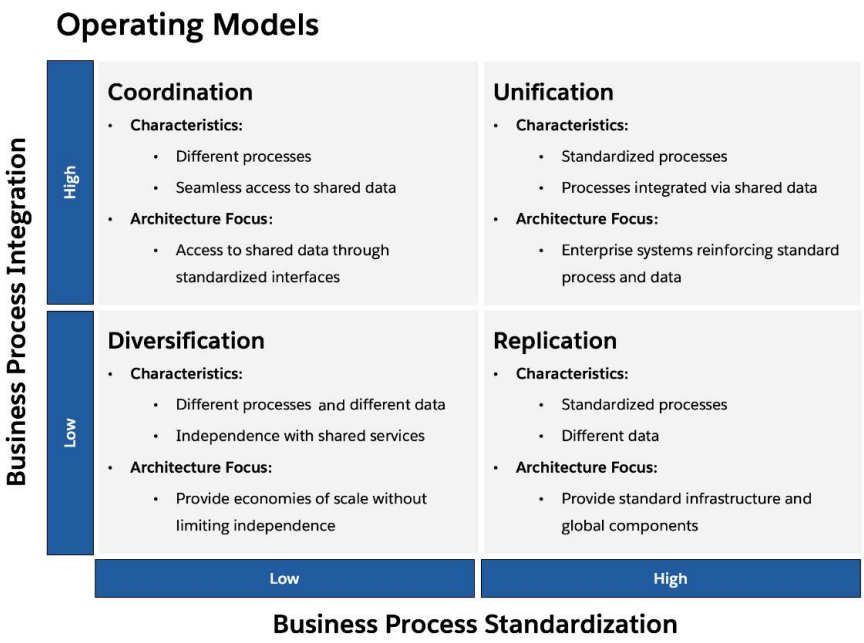


Figure 2.8 – SOGAF models

# Links

SOGAF framework that can be found on the Salesforce Architects site:
https://architect.salesforce.com/govern/operating-models/sogaf-operating-models

# Chapter 3

## Figures

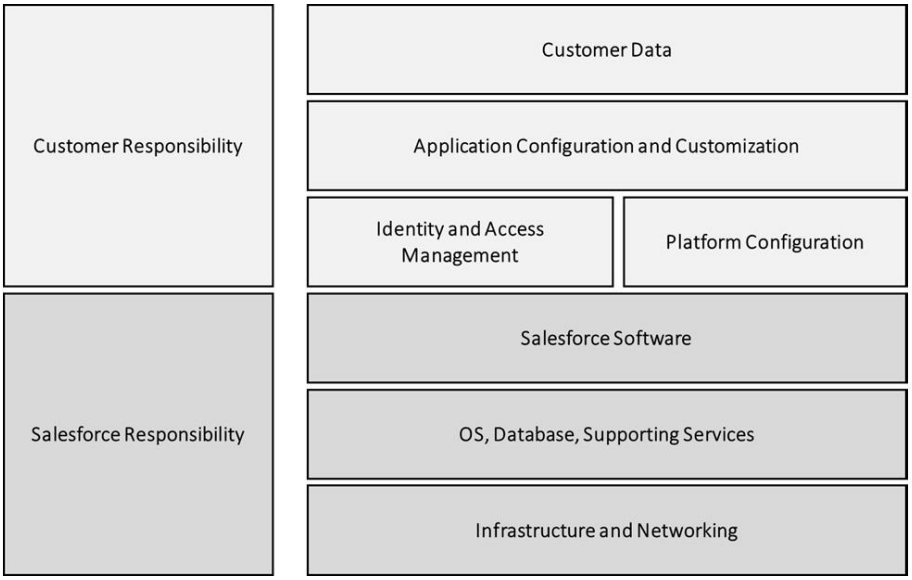


**Figure 3.1 – A shared responsibility model**

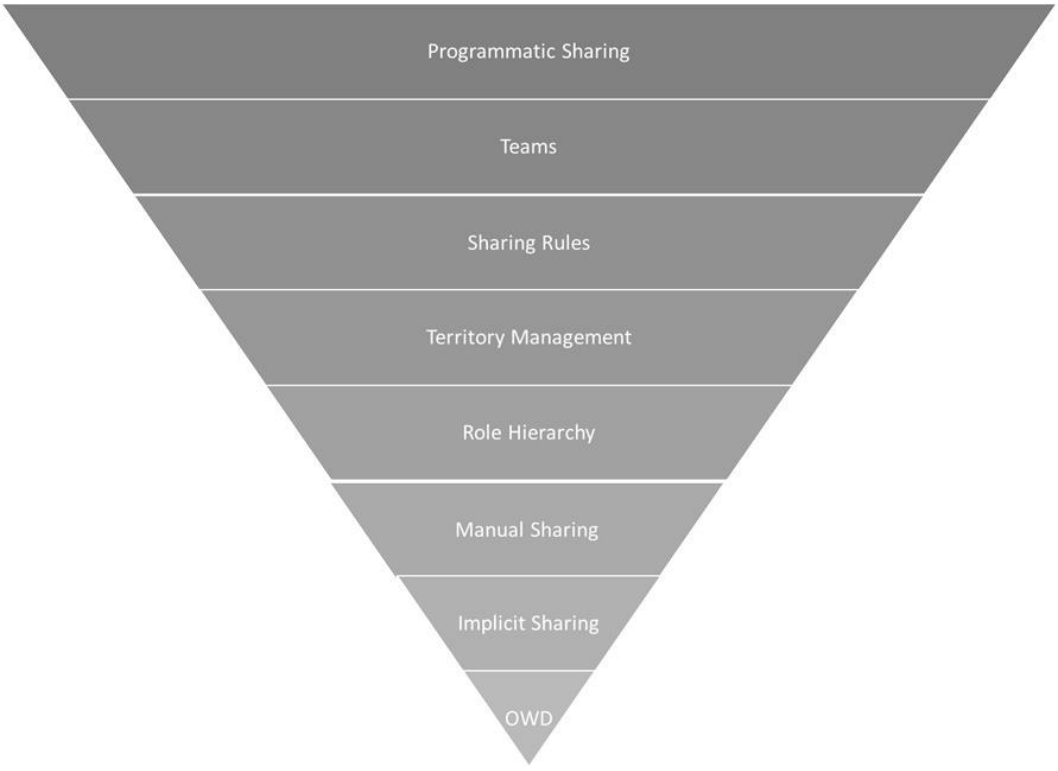


**Figure 3.2 – Sharing mechanisms**

# Table

| Mechanism | What it's used for |
|---|---|
| Ownership | Access to a record owner, user, or queue |
| Profile | Access to an object or not at the basic level |
| Permission Set | Access to an object or not at the basic level |
| Org-Wide Default (Internal) | Default internal sharing level |
| Org-Wide Default (External) | Default external sharing level |
| Role Hierarchy | Access to subordinates' records |
| Grant Access Using Hierarchies | Sharing with a role hierarchy or not |
| Public Groups | Sharing to a defined group of users |
| Ownership-Based Sharing Rule | Sharing based on a configured rule, based on record ownership |
| Criteria-Based Sharing Rule | Sharing based on a configured rule, based on record field values |
| Guest User Sharing Rule | Special sharing to the guest user |
| Account Teams | Sharing based on membership in an account team |
| Opportunity Teams | Sharing based on membership in an opportunity team |
| Case Teams | Sharing based on membership in a case team |
| Manual Sharing | Sharing records manually |
| Implicit Sharing | Automated sharing between parents and children for certain standard objects |
| Territory Hierarchy | Sharing based on configured territories and territory membership |
| External Account Hierarchy | Grant access based on account hierarchy |
| Sharing Group | Sharing records owned by portal users |
| Sharing Set | Sharing records to portal users based on matches between Account or Contact fields |
| Manager Groups | Sharing records with our management chain |
| Apex Sharing | Sharing programmatically |

Table 3.1

# Links

Understanding Salesforce Flows and Common Security Risks:
https://appomni.com/resources/aolabs/understanding-salesforce-flows-and-common-security-risks/

# Chapter 4

## Figures

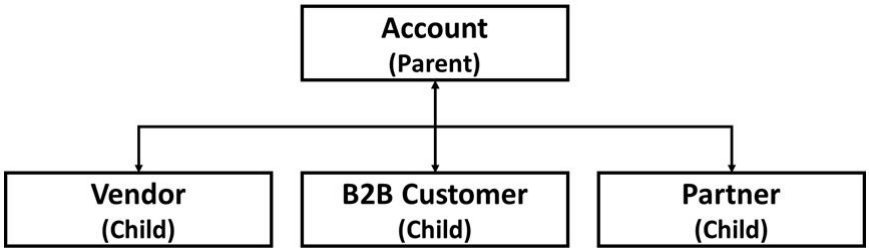


Figure 4.1 – Proposal for refactoring account model

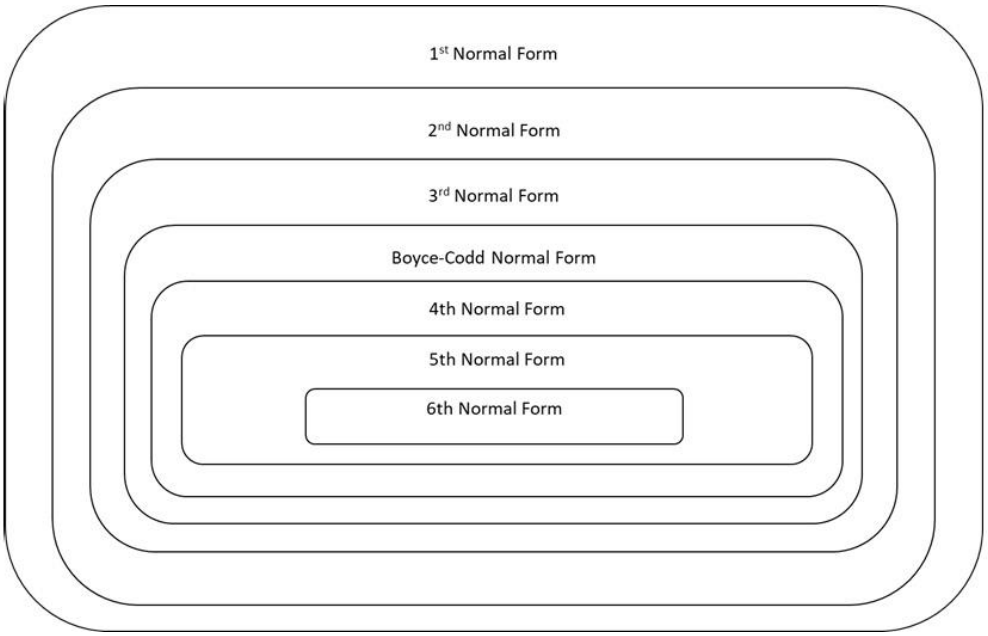


Figure 4.2 – Hierarchy of normal forms

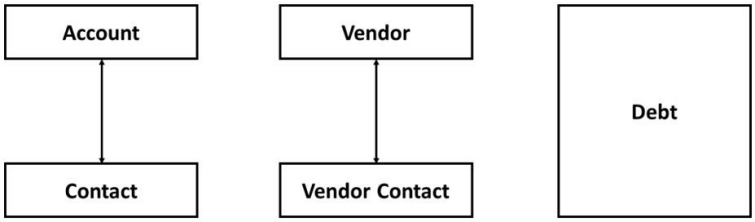


Figure 4.3 – Three different ways of representing account and contact

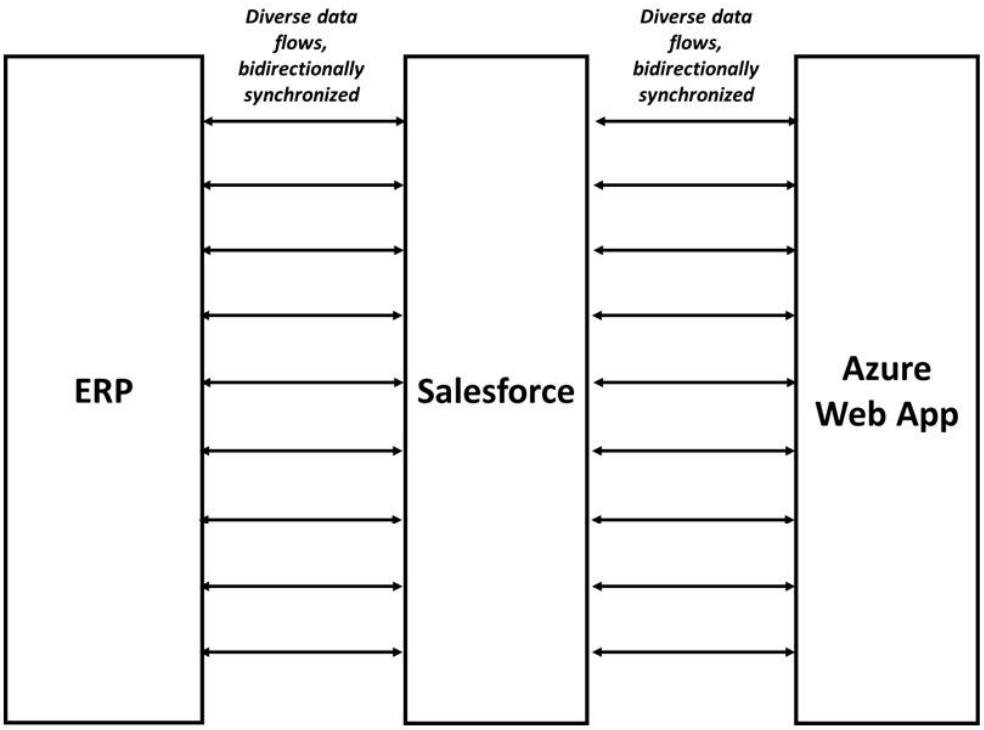


**Figure 4.4 – BusyCo synchronization scenario**

# Chapter 5

## Figures

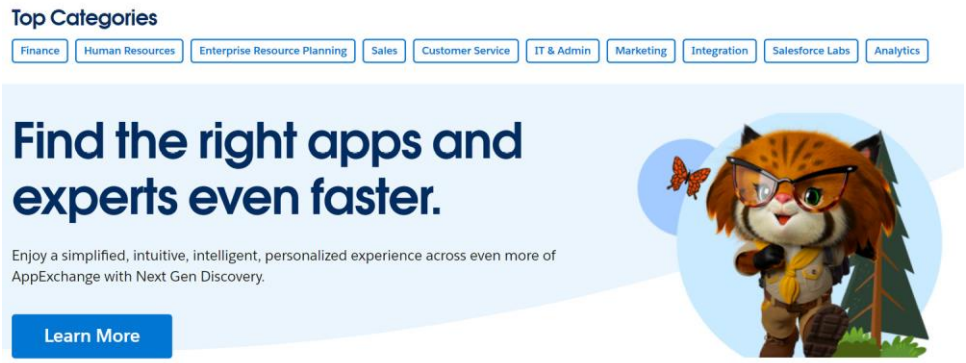


Figure 5.1 – AppExchange Top Categories section

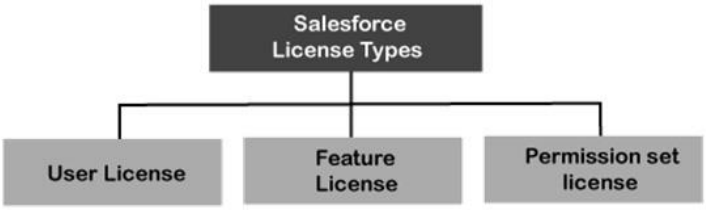


Figure 5.2 – Salesforce license model


Figure 5.3 – Scale of deteriorating results for assumption-driven customization


**Figure 5.4 – Some Salesforce automation options**

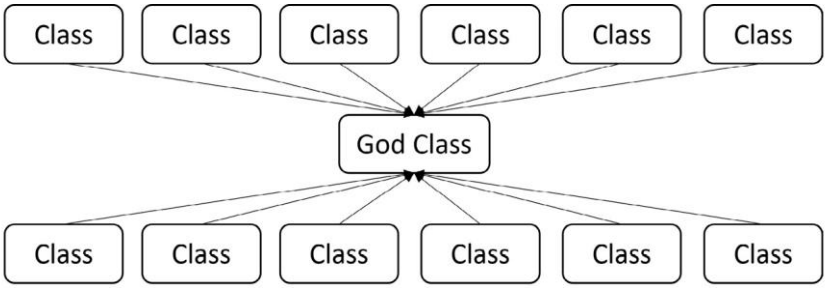


**Figure 5.5 – God class code structure**

```
public class OpportunityAddController {
        public Opportunity newOpp {get; set;}

        public OpportunityAddController(){
                newOpp = new Opportunity();
        }

        public void addOpp(){
                try{
                        insert newOpp;
                }
                catch(Exception ex){ }
        }
}
```

**Figure 5.6 – Simple example of error hiding**

# Links

Apex Trigger Actions (https://github.com/mitchspano/apex-trigger-actions-framework)

Nebula: https://github.com/jongpie/NebulaLogger

# Chapter 6

## Figures

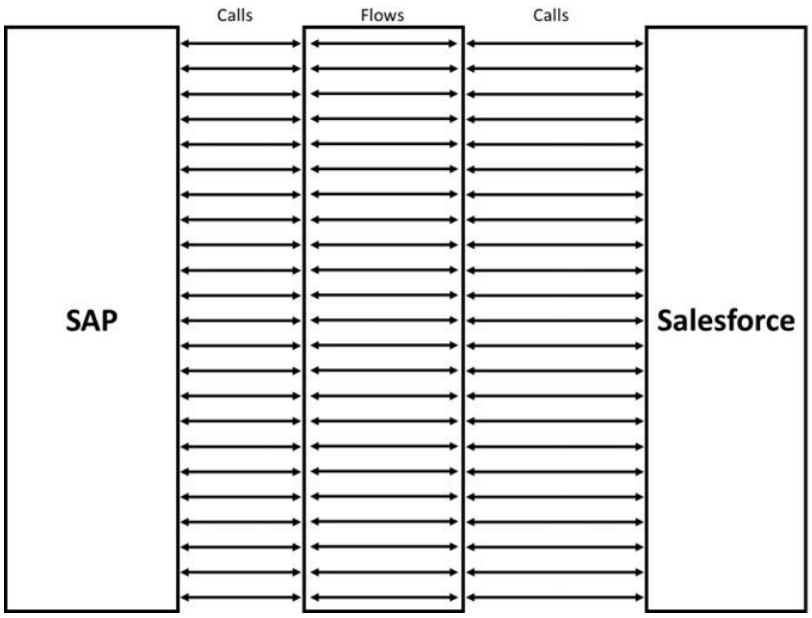


**Figure 6.1 – PumpCo integration architecture**

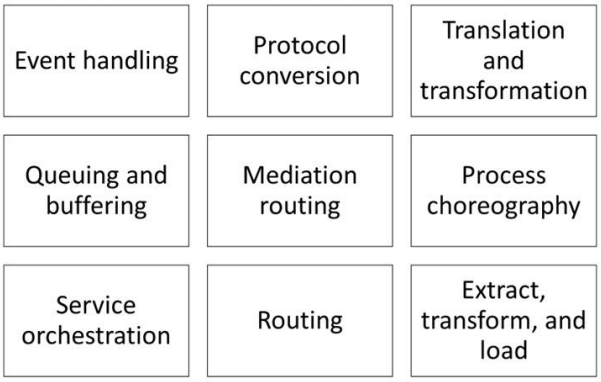


**Figure 6.2 – Common middleware capabilities**

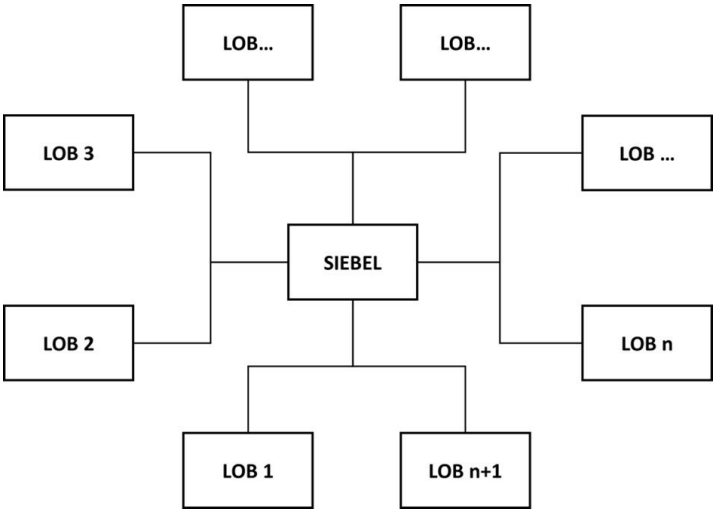


Figure 6.3 – Old Siebel setup at OmniCo

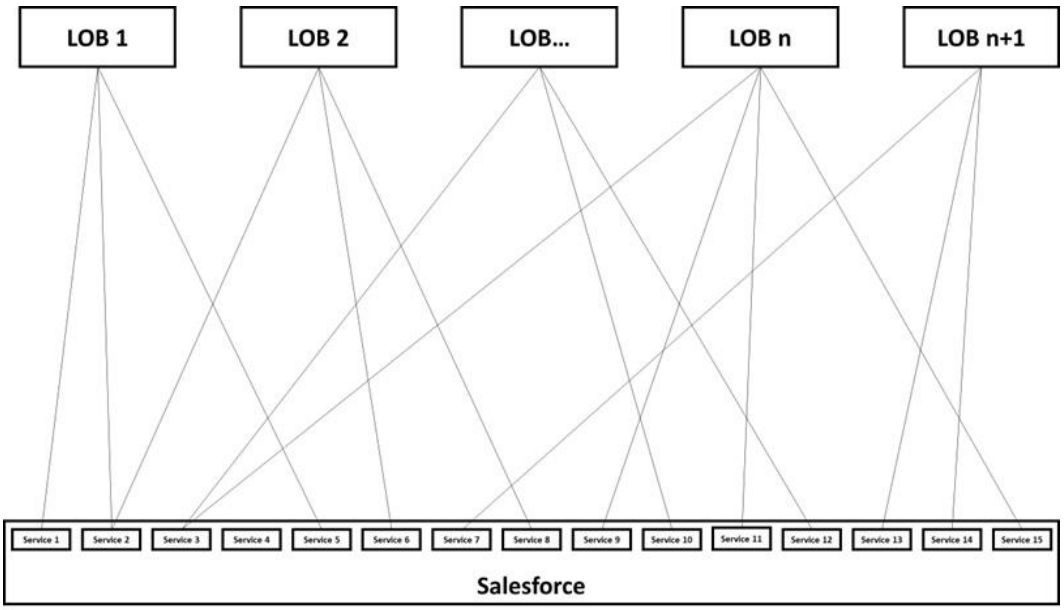


Figure 6.4 – Schematic view of OmniCo Salesforce integrations, not including ESB

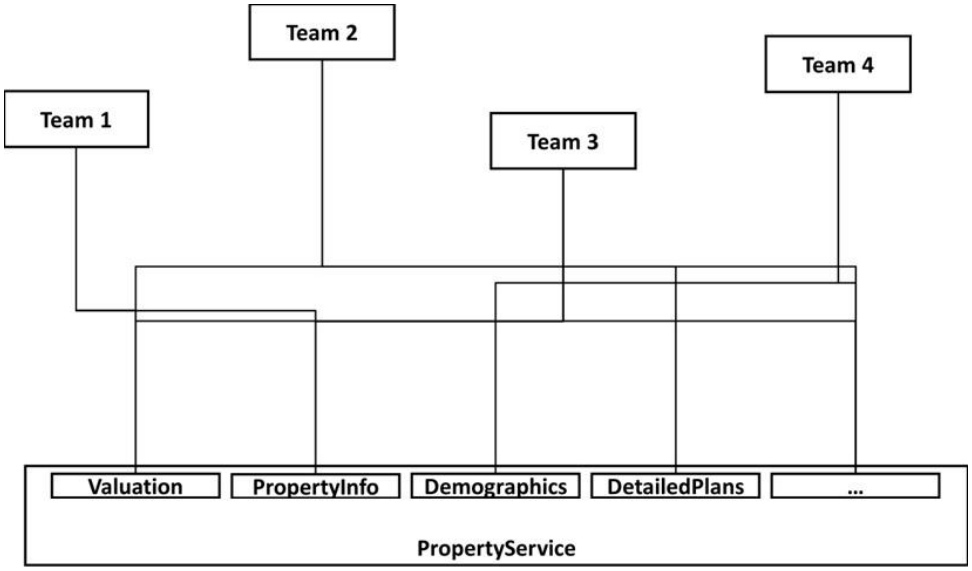


Figure 6.5 – View of RealCo PropertyService

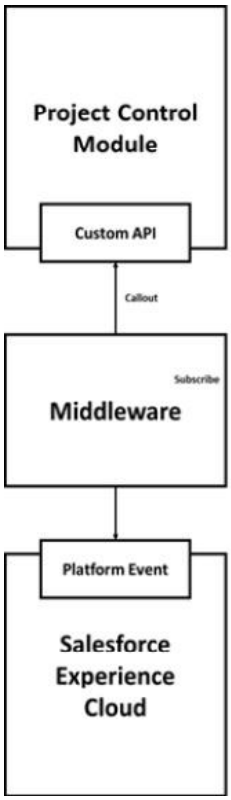


Figure 6.6 – WoodCo integration architecture

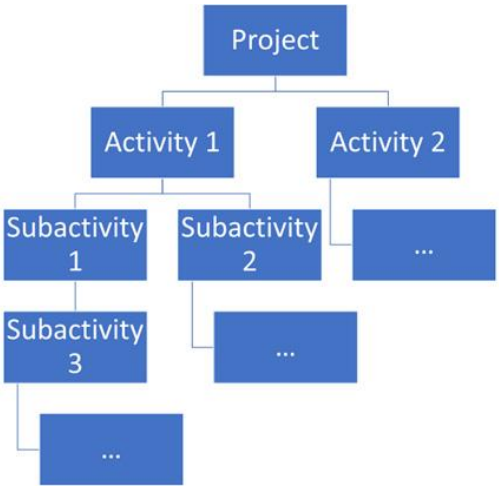


Figure 6.7 – WoodCo project structure

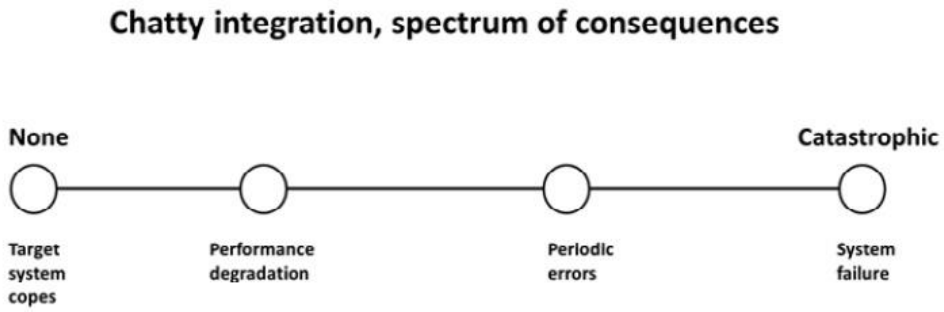


Figure 6.8 – Chatty integration spectrum

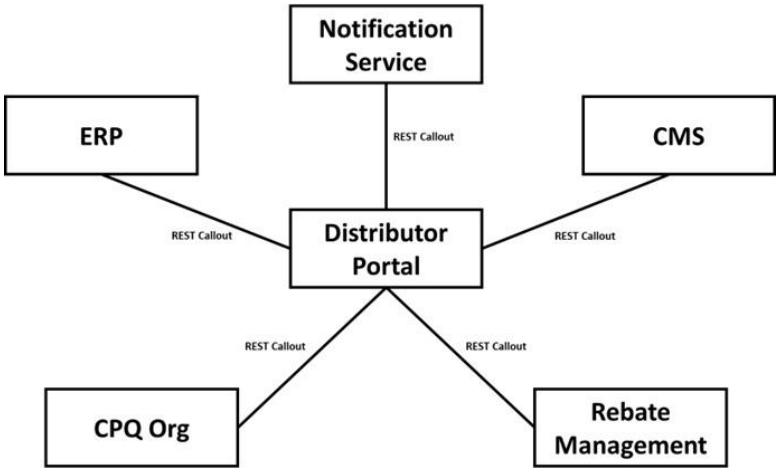


Figure 6.9 – Proposed integration architecture for WineCo

## Integration from Salesforce

| Type | Timing | Preferred Pattern |
|---|---|---|
| Process Integration | Synchronous | Remote Process Invocation—Request and Reply |
| Process Integration | Asynchronous | Remote Process Invocation—Fire and Forget |
| Data Integration | Synchronous | Remote Process Invocation—Request and Reply |
| Data Integration | Asynchronous | UI Update Based on Data Changes |
| Virtual Integration | Synchronous | Data Virtualization |

## Integration to Salesforce

| Type | Timing | Preferred |
|---|---|---|
| Process Integration | Synchronous | Remote Call-In |
| Process Integration | Asynchronous | Remote Call-In |
| Data Integration | Synchronous | Remote Call-In |
| Data Integration | Asynchronous | Batch Data Synchronization |

**Figure 6.10 – Overview of Salesforce integration patterns**

# Chapter 7

## Figures

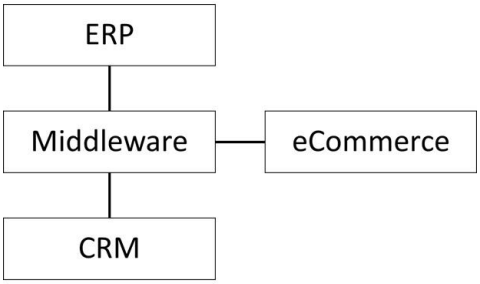


**Figure 7.1 – RollerCo new platform**

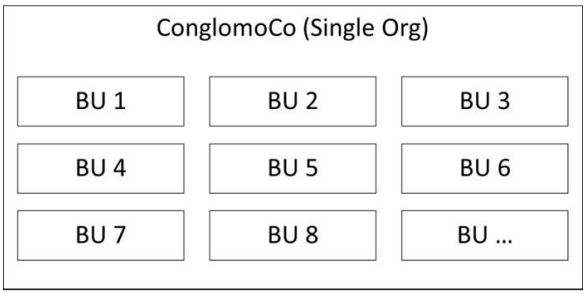


**Figure 7.2 – ConglomoCo org**


**Figure 7.3 – Package structure**

## Code

### Code 1.1:

Example of a dummy test in the following code snippet:

```
@isTest
class dummyTest{
  static testMethod void notRealTest(){
     //assume A is a class with two methods
     A aInstance = new A();
     aInstance.methodA();
```

```
      aInstance.methodB();
      //nothing is done to test anything
  }
}
```

# Chapter 8

## Links

Salesforce Diagramming Framework: https://architect.salesforce.com/diagrams

# Chapter 9

## Links

Anti-Pattern: https://wiki.c2.com/?AntiPattern

DevIQ has an excellent repository of anti-patterns that discusses many of the common ones that can be found across platforms, https://deviq.com/antipatterns/antipatterns-overview