Adaptation in the Open Framework

SyncHR’s Software as a Service (SaaS) applications are built on an Open Source framework we call the Open Framework. This framework makes extensive use of Object-oriented principles to simplify setup and operations. This paper discusses how these Object-oriented principles make adapting applications to company requirements easier.

Introduction

In order to provide the adaptability that sophisticated companies require in their HR application, various approaches have been used.

Parametric Approach

The application has a multitude of parameters that are filled in to configure the application to a company’s needs. This approach leads to cumbersome setup since many parameters need to be filled in. It also tends to make the application brittle since the code must consider all the parameters.

Code-based Approach

The application provides a programming language like Java or a proprietary 4GL language for modifying the behavior. This also leads to cumbersome setup since it requires technical skills. The approach also leads to significant upgrade issues since each company has a different code base.

Object-oriented Framework

With the advent of Object-oriented systems, another approach to adaptation is available:

Object-oriented Inheritance. While there are numerous advantages to using an Object-oriented design, Inheritance has enormous power in the SaaS environment.

Inheritance

The basic idea of Object-oriented Inheritance is that we have a basic facility (Object), say a Person. It has attributes like Name, Address, and so forth. This Object is available to every company. A company needs to track Licenses. The Person Object can be extended with License attributes without having to modify the basic Person Object. Every company can continue to use the Person Object while the one company uses Person with the License extensions.

SyncHR’s development approach is to build robust applications that have most of what companies need and then extend the application as new requirements emerge. Most extensions are simply the addition of new functionality. Those extensions can become available to any company by simply opting in. Individual company requirements can be implemented as extensions without modifying the common code base.

Structural Inheritance

By using Structural Inheritance SyncHR applies Inheritance to the SyncHR Synchronicity Model. (The Synchronicity Model is the underlying data model.)
Real world example:

A company has several subsidiaries. It also has a set of benefit plans that are shared among the subsidiaries. However one subsidiary was the result of an acquisition and has a separate suite of benefits.

Certainly the code that determines the available plans for a person could be modified to consider the different companies and modified again for the next acquisition. Or one could copy the plans from one company to another and be sure to copy any subsequent changes.

With Structural Inheritance, any company’s benefit plans can be inherited from another company. The subsidiaries that have the same benefits as the primary company use a company relationship to specify the benefits inheritance profile.

At Open Enrollment or a Life Event, employees are presented with the suite of benefits appropriate to their employment company.

Inheritance provides another powerful tool: Override.

If one of the subsidiaries has an additional plan beyond the primary suite of plans, that plan can be added as an Override. The plan profile available to that company's people is the inherited plans with the local extension.

Process Inheritance

In the Open Framework, processes such as the Payroll Calculation, Workflow, and the Benefit Event Calculation are Objects. Since they are Objects, Object-oriented principles apply. Here’s how Inheritance applies to these processes:

Pay Processing Example:

A company has an incentive plan that uses group performance information and a complex formula to calculate a quarterly bonus.

The group performance information is added to the Organization Object as an extension. The Employees have relationships to Organization as part of the core Object.

A Payroll Calculator is added to the Pay Process for the company. This calculator will be invoked only for this company as part of the payroll process. Adding the calculator involves no changes to the core Calculator. The rest of the Payroll Calculator functions use the common code.

When the Pay Process is used and the pay period is a quarter end, the additional pay records are written by the Extension.

Workflow Example:

The common compensation change workflow sends a merit change to the employee's executive manager. However, in one division of a company, they want the budget manager to be a reviewer.

A new role definition for the company is added as an Override to the role definitions. Since role definitions are company relationship based, it only applies to this company.
**Game Changer**

Using the Object-oriented principle of Inheritance makes it feasible to construct robust applications that can be used by a variety of companies where each company sees the application just the way they need.

The Object-oriented adaptation process is not nearly as cumbersome as Parametric or Code-based approaches. The normal implementation process is to load data and then use the common Object as a prototype asking, ‘What would we like to be different?’ then Override or Extend as needed.

Object-oriented Inheritance allows a SaaS application to provide Enterprise level functionality without Enterprise level costs. A full-featured application can be made to operate just the way each company needs by simple adaptation methods that fit mid-sized pocket books.