



White Paper

## APPLYING LOOSELY-COUPLED ARCHITECTURES TO THE CONTENT DRIVEN ENTERPRISE

*How Loosely-Coupled Architectures deliver flexible  
Component Content Management Solutions to the Enterprise*

Table of Contents

Executive Summary.....3  
High Cohesion, Low Coupling .....3  
Increase Technology Freedom .....5  
Specialization and Division of Labor .....5  
Increased System Scalability.....6  
Increased Reliability .....6  
Eliminate Vendor Lock-in.....6  
The SR2 Way .....6  
Summary .....7





## Executive Summary

When investigating the options for a Content Management System (CMS), it often seems that proprietary systems are the only option, and in the past, that's been true. The problem at that point then becomes finding a proprietary system that meets the needs of the business users within your organization, while hopefully minimizing the inevitable change to your architecture. However, if you dig a bit deeper, you may realize that the options that exist today fall into one of two categories, proprietary end-to-end systems, and Component Content Management™. The goals are still the same: A system that meets the needs of your business users while finding something that works within your existing infrastructure. The manner in which to reach these goals however, can differ greatly.

Proprietary end-to-end systems often address the Content Management problem in a one dimensional way. Often the solutions consist of a content creation/editing module; a complex, proprietary storage mechanism; and a proprietary publishing paradigm. The result: A website in a box.

Component Content Management solutions however, look beyond the creation of a simple website to focus on the concept of the Content Driven Enterprise; driving multiple sites, and internal and external communication, globally. More importantly perhaps, these solutions shun the proprietary means of storage and publishing in favor of open standards that will not only minimize the change to your architecture, but may eliminate it entirely, allowing the enterprise to leverage its pre-existing assets and infrastructure.

Component Content Management does this by breaking apart the process of content management and applying a high cohesion, low coupling approach to the solution.

## High Cohesion, Low Coupling

In software; coupling, or dependency, is the degree to which each program module relies on each other module. Coupling can be "low" (also "loose" or "weak") or "high" (also "tight" or "strong"). Low coupling, or a loosely coupled architecture, is the preferred method of implementation, in that each module can stand on its own, with minimal inter-dependencies, providing an extremely high level of reuse, and freeing the developers to quickly link components together to assemble a variety of business solutions. Low coupling is a sign of a well structured computer system.

Coupling is usually contrasted with cohesion. Low coupling often correlates with high cohesion, and vice versa. Cohesion is a measure of how well the code within a module works to provide a specific piece of functionality. Simply put, Cohesion measures how well a module does the one thing for which it was designed. Cohesion is usually expressed as "high cohesion" or "low cohesion". Modules with high cohesion are considered preferable because high cohesion breaks the problem into



pieces; each module is laser-focused on solving a single area of the problem. This approach is usually associated with several desirable traits of software including robustness, reliability, reusability, and understandability. Conversely, low cohesion looks to solve the problem as a whole, and is associated with undesirable traits such as being difficult to maintain, difficult to extend, difficult to test, difficult to reuse, and even difficult to understand. The goal here again, is that cohesion is focused on executing a specific task. Multiple high cohesion modules may manage a very complex task, while a single low cohesion module would manage the same task. The difference being the ability to maintain, update, or scale that task is much easier with a high cohesion framework, where segments of the process can be updated without re-writing the entirety of the process.

By combining the two concepts, and creating loosely coupled, high cohesion program modules, an organization can limit programmatic inter-dependencies, and a library of proven, defect-free, highly-reusable components can be achieved. Additionally, by varying the combinations of these components, a multitude of business solutions become possible, without the need for custom coding. The key to building a High Cohesion, Low Coupling architecture is to ensure each component of the architecture has a level of independence, and a well defined interface to foster component integration.

The benefits of a High Cohesion, Low Coupling architectures include:

- Component Optimization
- Component Specialization
  - o Each component can be developed, maintained, updated and managed by different teams. This aids division of labor, project management and isolation of roles, responsibility and concerns.
- Component Independence
  - o Each component can be developed independent from the system and in the most appropriate programming language, using the most appropriate hardware and operating system. This aids division of labor, project management and isolation of roles, responsibility and concerns.
- Loosely coupled systems are inherently more open, since the points of coupling are well-defined, and system transparency is increased.
- Decreased dependencies increases system uptime and reliability
- Scalability
  - o Best of breed components can be interchanged and updated without impacting the system
- Flexibility
  - o Avoid vendor lock-in



## Increase Technology Freedom

There are many CMS providers who offer tightly coupled all-in-one "solutions," with a single monolithic system providing editing and archival functionality for the technical staff, and a front-end website application for use by the outside world. These systems do an adequate job of filling specific niches in the online ecosystem, by allowing companies to produce regularly updated websites. Companies with simple needs may find this kind of architecture adequate. However, companies who desire to make sites that go beyond what the other players in their field are doing, those striving for the Content Driven Enterprise, will find these "website in a box" solutions inadequate. Refresh Software is dedicated to delivering the tools necessary to achieve the Content Driven Enterprise, and does so with SR2, the company's loosely coupled, high cohesion solution that provides the best possible content management environment for developers, system engineers, content stakeholders and business advocates.

By applying the tenets of high cohesion and low coupling to the entire process of Content Management, as well as its technology, SR2 has broken apart the Content Management process, eliminating the dependencies from one phase of the process to the others, and creating independent modules that can be combined to create a unique content management solution. In doing so, Refresh has opened the doors for technological freedom, providing the ability to leverage best of breed storage and publishing tools within the Component Content Management solution.

## Specialization and Division of Labor

With a tightly coupled system, your company is tied to the procedure the CMS vendor envisioned. Most CMS vendors offer an extremely limited range of languages and technology platforms. In order to utilize these systems in new and interesting ways, a company's technical staff is compelled to become experts in the platform and languages in which the CMS has been written. This is, of course, assuming the core system was designed to be expandable at all. Conversely, a component content management system based on open standards, allows the enterprise to create compelling applications on the front end, utilizing whatever platforms and languages they see fit, while delegating the content management tasks of storage and publishing to best of breed DB experts, and publishing tools. The component content management approach, like the concept of high cohesion and low coupling, allows for laser focused specialization and a division of labor within the enterprise, providing the corporation with the ability to utilize the most appropriate technology and resources to address the challenges at hand, all resulting in a more robust solution with faster time to market.



## Increased System Scalability

With a monolithic, highly coupled system, options for scaling are often limited to “throw a bigger server with more memory” at it. This can be a prohibitively expensive proposition for companies on the rise. With a singular point of coupling between the CMS backend and the outward facing front end, however, either side can be expanded according to the company’s needs. A typical expansion pattern might be adding a cluster of inexpensive servers to host the front-end application while keeping the backend application on a single machine. Additionally, with the componentized approach to CMS that Refresh Software provides, organizations are not limited to specific proprietary hardware, but can place the challenge of scalability firmly in the hands of the experts, by utilizing best of breed storage solutions within their content management process.

## Increased Reliability

While nearly every system will experience downtime if the point of coupling fails, a tightly coupled monolithic system is by its nature more prone to massive, system-encompassing failures. A computationally intensive task (such as archiving or large scale data-import) on the backend puts the performance of the entire system at risk. If something “catastrophic” happens to the backend, the front-end will go with it, as the entire system lives and dies as one.

A loosely coupled architecture on the other hand, increases system reliability and uptime by reducing single points of failure, and breaking apart the interdependencies of a tightly coupled system.

## Eliminate Vendor Lock-in

CMS vendors are interested in keeping your business. One tactic is to keep your data in a closed proprietary system. If a company decides to change vendors, data migration then becomes a difficult task. Similarly, with a tightly coupled system, the front-end application will often have been written in the same technology as the backend system. A less scrupulous vendor may insist on large quantities of expensive consultant time in order to teach the ins and outs of their tightly coupled systems (assuming that the front end has been designed to be open enough in the first place) to allow the front-end to be re-usable in any form.

A loosely coupled system has well-defined, published, and easy-to-use interfaces that allow for easy incorporation to existing or new front end applications.

## The SR2 Way

The SR2 system is an excellent example of a loosely-coupled system. For publishing there is one point of interaction between the SR2 application (a powerful J2EE engine allowing creation, editing, and archiving of all manner of corporate intellectual assets) and the front-end applications that allow controlled access to the outside world: the database. An

implementer of the front-end application uses the programming language and toolsets that already know to leverage the SR2 Content Retrieval API (current implementations include .NET, classic ASP, Java Struts/JSP Taglibs, ColdFusion, PHP and Perl) or communicate to the database directly. This "platform agnosticism" (allowing the developers to use the toolsets they are most comfortable with) even extends to the database, with support for MS SQL Server, Oracle, and MySQL.



System developers often have the need to work with other groups or staff, often non-technical, to define the Content Object Model of the system. This Content Object Model, the outline and form of the data that the system needs to manage, as well as the workflows that content must pass through for approval before it is published to the outside world can be created to meet the exacting needs of the organization. Additionally, the two groups may collaborate on the templates the front-end system utilizes to display the data, the look and feel of the site. Once these are defined, the content staff will login to the SR2 system (a tested, scalable J2EE application) and edit the data. An optional point of coupling between the backend and front-end is the preview function, where the SR2 application may utilize templates on the front end in order to confirm the appearance of newly created or edited pages.

It is important to note that content consuming applications are not limited to traditional dynamic web applications. Refresh Software's loosely coupled architecture has allowed companies to publish their data as static websites, PDF, Word Doc files, and graphics. Many of these extensions would be difficult or impossible in a tightly coupled, monolithic system.

## Summary

At the macro level, companies are given maximum flexibility to get their independent systems to interoperate. At the micro level, programmers find decoupled architecture the key to reusable and flexible code, maximizing technical skill sets, leveraging existing investments in infrastructure, and minimizing risks associated with vendor lock-in. SiteRefresh has been built on these same principles, and demonstrates the increases in freedom, flexibility, scalability, and reliability that this architectural pattern provides. SiteRefresh has been designed to work within an existing information infrastructure, to leverage best of breed components, and to provide unparalleled freedom for implementing a content management solution with the tools and resources selected by the enterprise.