DEVOPS FOR THE ENTERPRISE: ALIGNING AND STRUCTURING PROJECTS, PART 1

Patrick McClory July 7, 2016 DevOps 6,443 Views

Previously, we discussed the shift in moving to a DevOps-oriented view of system development and management. The next concern when enabling the DevOps workflow is organizing the work in a way that is manageable and is sufficiently discreet to enable end-to-end ownership by the involved teams. As a rule of thumb, modules that are partitioned should encapsulate a fully functional process or set of processes. By doing this, the question of ownership over any given module or set of modules will be removed as teams will be assigned at a module level.

Application design best practices include deployment management as well as:

- Automating everything from continuous integration to deployment and fail-back processes
- Loosely coupling disparate systems by standardizing on the design and protocol first
- Utilizing rate-limiting processes within service registration and management workflows

- Managing infrastructure tasks in parallel processes as much as possible to speed deployment and recovery processes

Let us discuss a hypothetical customer relationship management (CRM) package as a way of looking at restructuring an application for this type of team workflow. For simplicity's sake, let us assume a simple system that tracks customers, captures contact history data, a generic set of marketing tasks along with some low-level logging and application instrumentation tasks.

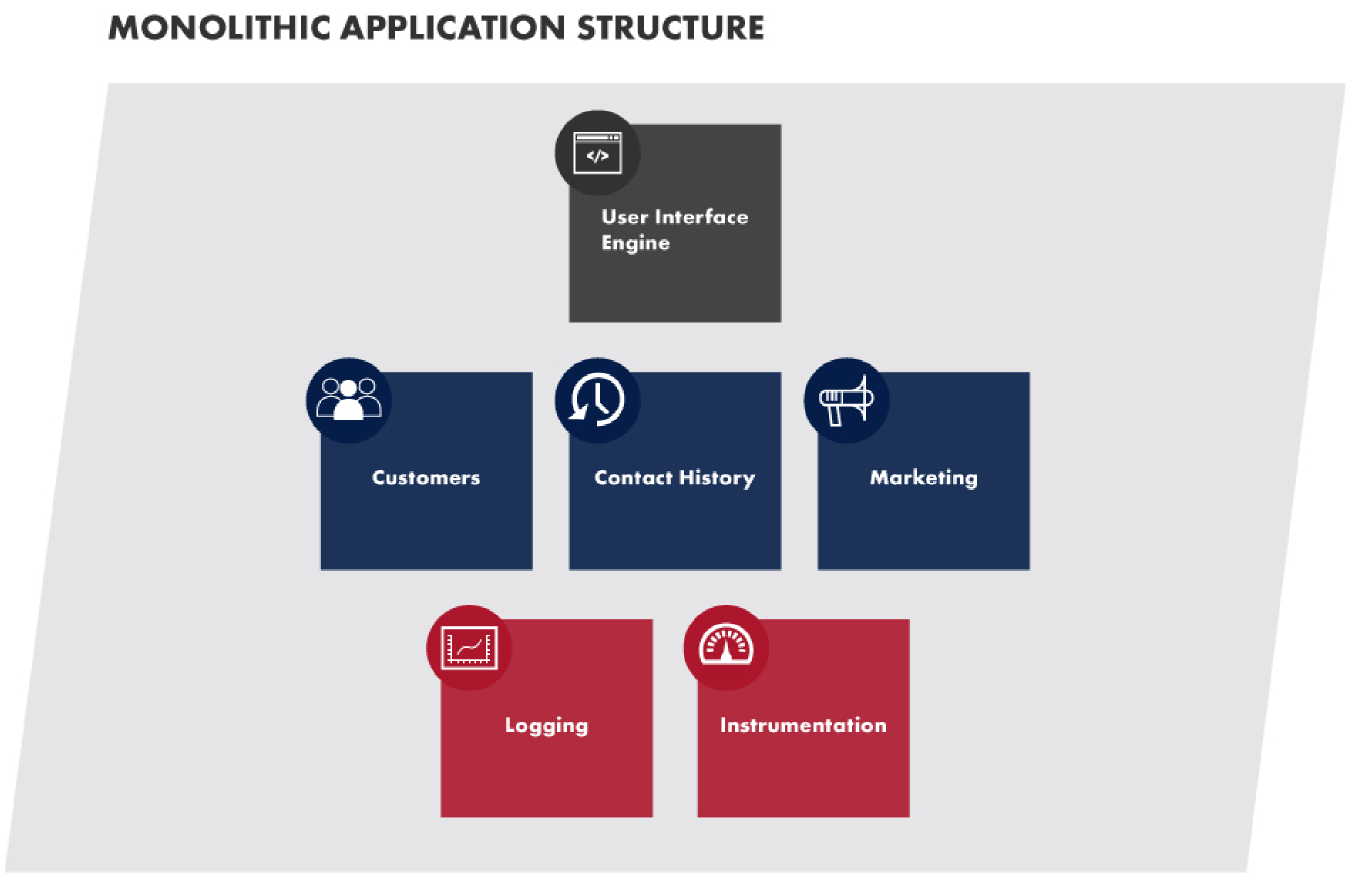

Figure 1 – Monolithic application structure

An easy first decision in this case would be to break the system up by the three layers: user interface as the top layer, business objects in the middle layer, and a common set of low-level tools common to all projects on the bottom layer. While there is benefit to decomposing the layers of an application in this way, aligning teams to this structure is still difficult in larger applications. Additionally, changes in one area, at a minimum, could require a disruption in other areas of the system for redeployments and other updates to the infrastructure. Further decomposing the system into smaller components can afford greater flexibility in team alignment and ownership as well as greater deployment stability. Rather than being tightly coupled and bound within a monolithic architecture, modular components offer a potential for becoming more fault tolerant by designing components to handle failure gracefully. Further, cascading failures within a modular design, so long as the dependency tree of the components is well designed, do not necessarily have to result in a full disruption of service.

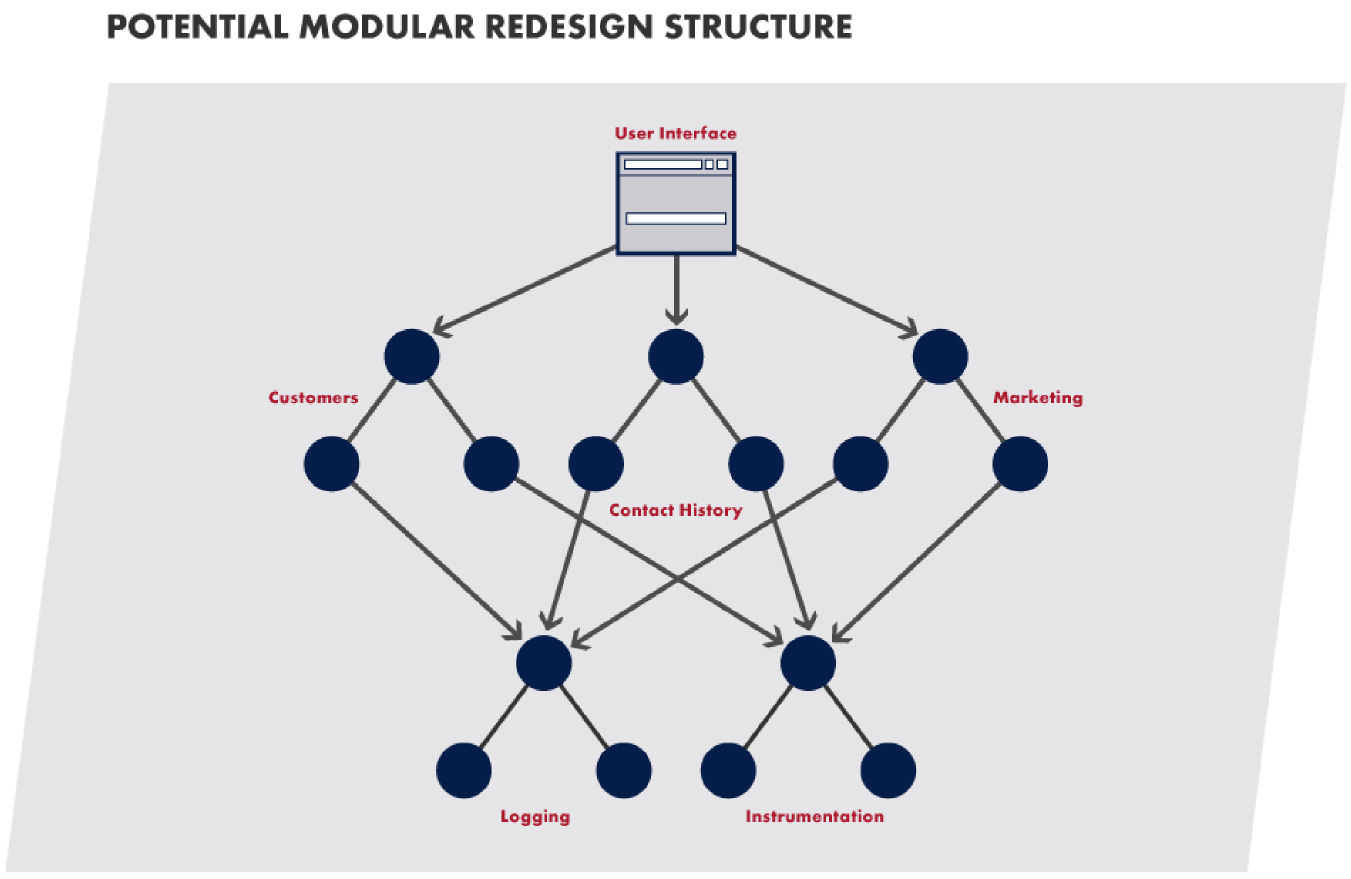

Figure 2 – Potential modular redesign structure

While the example above describes a simplistic, legacy application, using the overall concept of modular design allows for deployment and infrastructure management concepts to influence the design and development of systems. It also offers the opportunity to inject infrastructure-as-code into the workflow within the development effort. While not a requirement or the only pattern implemented with DevOps teams, Service Oriented Architecture (SOA) tends to be a popular approach for teams that are working toward building a cloud-focused architecture strategy. This encapsulation method also serves well as the basis for such a design paradigm. Pairing deployment and infrastructure code with each service neatly encapsulates the entirety of the lifecycle of that service in one place.

That brings me to collaborative automationThe intersection of software development and IT operations exemplifies the opportunity that exists for cross-pollination and the types of creative solutions that the separate IT disciplines can identify and build upon when they share ownership of a product from end-to-end. With the technical work organized to fit with the structure of the teams working on it, automating the process of managing the end-to-end process from check in through environment deployment and testing is critical. While automation in the IT space is not a new concept at all, using consistent techniques across development and the production environment is certainly a rarity and an area where universal needs of the DevOps community have yielded a number of new automation tools in the commercial and open source markets that operate at scales previously unmanageable. The catalyst in the change of mindset and approach was the concept of looking at infrastructure automation process as code and using well-known processes that have been finely tuned by the software development community such as source control management and continuous integration.

Automating processes across multiple IT disciplines is key and it reduces the amount of overhead and human interaction required to manage the application lifecycle. While it's certainly helpful to automate a process to build and package an application on a regular or continuous basis, a build that has not been tested (unit, functional and deployment testing) has little value by itself. To accomplish this, it is necessary to include a few specific (such as infrastructure as code and configuration management) types of tools to manage and orchestrate the process. From a strategic perspective, many teams opt to repurpose tools they already have or look to open source toolsets as a means of saving money and providing a flexible set of tools that can be extended or customized on demand. Certainly teams that are not currently implementing this level of automated testing can layer it on if and when it becomes appropriate.

More on the alignment and structure of projects next week, when we dive into additional components in the suite of important DevOps tools.