CiRM Application Portfolio Catalog

# TOGAF Preliminaries

The purpose of this catalog is to identify and maintain a list of all the applications in the enterprise. This list helps to define the horizontal scope of change initiatives that may impact particular kinds of applications. An agreed Application Portfolio allows a standard set of applications to be defined and governed. The Application Portfolio catalog provides a foundation on which to base the remaining matrices and diagrams. It is typically the start point of the Application Architecture phase. Existing application registries and repositories — such as SAP’s Solution Manager and System Landscape Directory products — also provide input into this catalog from a baseline and target perspective.

The Application Portfolio catalog contains the following meta-model entities:

* **Information System Service -** a thing that a business does that has a defined, measured interface and has contracts with consumers of the service. Information system services are directly supported by applications and have associations to SOA service interfaces.
* **Logical Application Component -** a class of application. An encapsulation of application functionality that is independent of a particular implementation. For example, the classification of all purchase request processing applications implemented in an enterprise.
* **Physical Application Component**- an actual application - An application, application module, application service, or other deployable component of functionality. For example, a configured and deployed instance of a Commercial Off-The-Shelf (COTS) Enterprise Resource Planning (ERP) supply chain management application.

**Notes:**

* The TOGAF process suggests that the application architecture be independent of specific technologies. However, it is nearly impossible to talk about a solution to a problem without talking in terms of the solution domain (i.e. specific technologies). That said, I've tried to avoid reliance on any specific product or non-standardized technology.
* TOGAF's approach seems targeted towards a more vertical partitioning of the application architectural components. However, the CRM architecture envisioned in this document relies heavily on a model-driven approach where a handful of generic components implement nearly all business logic based on declarative models describing the business domain in terms of entities and rules.

# Architecture Strategy

The architecture we are adopting is a model-driven architecture based on semantic web standards such as OWL and related specifications. We strive to define as much of the application logic, including user interfaces in terms of executable models: ontology, metadata, rules and workflows . Nonetheless, we leave the possibility open for a conventional programming to take over when functionality cannot be covered in a natural way with models or when there is a significant performance improvement.

There are two broad categories of application components that make up the overall architecture:

1. **Horizontal components** forming the foundational infrastructure. Such components do not solve any particular business problem and do not cover any particular business functionality. Rather, they provide the runtime environment for a model-driven implementation of the business requirements. Most horizontal components are server-based. Some come in the form of libraries embedded within vertical components (e.g. a rules engine).
2. **Vertical components** covering specific business functions, domain dependent or not. Many vertical components are lightweight wrappers around generic models with customized business functionality. Many vertical components will be client-centric, fully relying on client-side technologies (such as Ajax and Flex) to implement business logic. Finally some of the standard services present in the enterprise such as email have been classified as vertical because they perform a well-defined, self-contained business function.

# Horizontal Components

## Workflow/Time Service

The Time Service triggers timely activities within the CiRM ecosystem. It is used to schedule and trigger background activities and real-time business rules for things like notifications and the handling asynchronous workflow processes. The Workflow Service coordinates business processes - it must support both workflow concepts such as tasks, state transitions, escalation, human activities etc. and service orchestration with mapping/manipulation of structured data to match published service interfaces. We must distinguish between scheduling and workflow execution. Off-the-shelf BPM servers will provide scheduling capabilities. If such an off-the-shelf is not used however, workflow execution can be separate from the scheduling service because scheduling services might be needed by components that do not directly participate in a workflow.

## Metadata Service

The Metadata Serviceis essentially an ontology database exposing a query and a manipulation interface. The service will be implement on top of an ontology database and it will offer a query interface (e.g. SPARQL) and a manipulation interface (to add/remove/modify ontological entities). This can safely be labeled the kernel of the whole architecture.

## Ontology Editor

The ontology editor will allow CiRM and KM (Knowledge Management) experts to manage the CiRM and content model. The ontology editor may in fact be multiple editors targeted to particular aspects of the ontology. However, the abstraction/modeling level on which the editor will operate remains OWL 2.0. More user-friendly administrative UIs that do not require semantic web expertise and that operate directly on the ontology, but are based on domain concepts (e.g. service request types and flex questions, rather than OWL classes and properties) will be developed as separate application components.

## Business Rules Engine

The business rules engine is a component working in conjunction with the metadata service. Business rules are going to be defined in terms of ontology metadata. Therefore, the natural choice of language is SWRL. The rule engine will have to support either SWRL, or a language that can be translated to from SWRL (e.g. SweetRule at <http://sweetrules.projects.semwebcentral.org/>).

The business rules engine is a component embedded in several services, including the Metadata Service, the Operations Service, etc. The business rules themselves, however, remain in the common metadata repository.

## Business Rule Editor

This could be a part (e.g. a plugin) of the ontology editor. Or it could be a plain text editor combined with a deployment and a change management process. The business rules editor will be used by programmers, or by technically savvy business users (a rare breed). Specialized business rule editors in the form of administrative UIs are going to be added on an as-needed basis. The generic editor should, however, be able to manage all business rules as they should all be translated into a single high-level rule language.

## Operations Service

Exposes a set of services in the form of HTTP REST interfaces and/or web services (WSDL) that implement business operations. It executes business rules and it writes to the operations database. The Operations Service is oblivious to user interfaces, user sessions, interactions etc. Each operation exposed by this service results in a persistent change and must leave the back-end data in a consistent state. In other words, each operation corresponds to a valid business transaction that, once committed cannot be rolled back.

This is pure business logic, some of it implemented on top of the rule engine, some of it coded up manually. It is accessed both by end-user components and the workflow engine. A custom dispatching mechanism internal to the server will route operations to the appropriate implementation.

## Operations Database

The Operations Database contains all operations data managed by the operations services. This database is the starting point for a reporting CRM data warehouse. Operations data is to be distinguished from metadata by the fact that the latter defines (and is essential to) application functionality while the former is not.

## User Service

Integrates all customer profile data into one view. Data will be loaded on-demand. Data will be merged from enet LDAP, portal LDAP, Bluebook, and other systems. Data will be searchable, where the search will work in an "on-demand" fashion as well: given what is known about a user, it will try to find a concrete business object in some system. All applications in the CiRM ecosystem will go through the User Service for any user related information, except authentication which will only be an optional part of this service. The User Service is also capable of managing user preferences on a per application basis.

## UI Service

Generates application user interfaces. The elements of a user interface are composed of GUI components. Such GUI components are going to be modeled in the ontology as “knowledge,” just like domain business objects are. The modeling is at a higher-abstraction level that permits rendering in different client-side formats. Examples of GUI components are the various standard input fields, forms, menus, tables and trees. The UI service will make use of rules to render a business object for a specific client-side engine. It is much easier to contextualize behavior along different dimensions such as client device, user role, workflow state etc. using rules rather than conventional imperative programming logic, because orthogonal aspects can be stated independently.

# Vertical Components

## Email Server

The email server will be used for outgoing notification emails. No particular extra requirements at this point. However, we still have the open question of tracking emails pertaining to a particular case. Perhaps some extra, proprietary feature of the email server could be use for that purpose (if available).

## Fax Service

The fax service is able to fax documents in one of standard formats that other application components would be able to easily produce.

## Chat Server

The chat server is a standard based P2P chat solution used for both constituent-county and county-county communication. Suggested product here is OpenFire, based on the XMPP standard. The chat server should allow one-to-one communication, chat rooms with roles and permissions. It should be possible to plug in and record chat sessions. (Note: OpenFire is already implemented in the PKBI/AnswerHub application).

## Chat Client Components

An AJAX and/or Flex chat client is available for mashups. The client should support person-to-person chats, chat rooms and conferencing chats with multiple *ad hoc* participants.

## Reverse 311

Automatically notify constituents by phone about certain events. It should be capable of text messaging as well, which should be exposable as a service accessible by CiRM workflows. This functionality is currently available through Communicator NXT and GeocastWeb (DCC’s Reverse 311 applications).

## GIS Service

Provides maps, address validation, jurisdiction determination, geospatial data, County-specific layers, routing.

## GIS Client

An interactive map display of GIS information that is browser embeddable: could be AJAX or Flex based. It must be embeddable within a mashup.

## Reporting Service

The Reporting Service provides reporting on operational data. Several reporting aspects need to be accounted for:

* Expert report definition - reports that require the design of data warehousing schemas and complex definition within the reporting software (Cognos).
* Custom/Simplified Report creations (custom, business specific interface?) - expose a simple interface for ad hoc reports .
* GIS based reports
* Report client display: graphs, numbers, tables (this is perhaps a separate service or a client-side library)?
* Real-time updates with server push - those will be really simple and predefined reports that are mostly likely to be coded manually. In general such real-time updates will avoid performing database queries, except at startup time. Rather, they will monitor appropriate events/messages/notifications and maintain the numbers in memory.

## Content Editor

The Content Editor manages all forms of textual content (externally available, on the intranet, KB etc.) The content editor produces artifacts (business objects) that follow a business process the same way any other business object (such as a service request) does. Content goes through a lifecycle: it is created, updated, published, unpublished, expired, revived, tagged, indexed, deleted.

The Content Editor is a full-fledged CMS (content management system) build on the CiRM infrastructure defined in this architecture. Content/media types are fully modeled within the ontology. Publishable artifacts, from informational snippets, to design templates, to complete web sites are also modeled in the ontology together with a set of business rules and workflows applicable to them.

The notion of content must be defined in the ontology with appropriate UI implementations - viewers, editors, permissions for the various types of content, workflow templates etc. The Content Editor is then a specialized user interface using those models.

## Keyword Text Search

An off-the-shelf keyword search engine customized to our needs. This search engine returns results very quickly and is heavily customized to perform well on particular keywords search combinations.

## Semantic Text Search

Ontology driven search using natural language processing that is able to provide harder to find information with a deeper analysis of the organization's content. This search engine is much slower to return results, but is more capable of matching a grammatical question (or a problem statement) with textual content available online. In addition, it exploits the entities and relationships defined in the ontology in a meaningful way.

## Virtual Assistant

The Virtual Assistant is an online autonomous software agent that is trained to have simple conversation with constituents via a chat interface. The goal is to answer common questions by using some pre-fabricated dialog scenarios. The Virtual Assistant will be able to connect to a real person if currently available. This is akin to automated phone customer service.

## Answer Hub Dashboard

The Answer Hub is the main application used by 311 information specialists. It is a mashup of several specialized applications, including but not limited to the following:

* search
* chat
* maps
* service entry
* soft phone

Those components are very tightly integrated within the Answer Hub, hence its status as a separate application component, unlike other mashups that are likely to be created in a more *ad hoc* manner.

## Phone Interactivity

The Phone Interactivity component is a call center solution that must have telephony integration points with the rest of the CiRM application sitting on a call taker's desktop. In particular, it must have soft phone capabilities, call transfer, voice recording and conferencing.

## Service Direct

A dedicated application for the intake and lifecycle management of service requests. Aggregates service request database management (listing of SRs, search of SRs with particular attributes), intake, service activities, reporting, constituent management - essentially what SVCDIR currently does, but with the addition of missing functionality such as the display of more detailed service request activities, updates, as well as some mobile/GPS-related shortcuts, such as automatic photo upload and automatic address population etc.

Additionally, Service Direct should be extended to IVR (Interactive Voice Response), which will be used to enable phone self-services, to offload call taker handling of user inquiries and transactions.

## User Admin

This application offers an administrative interface to manage full internal user profiles as exposed by the User Service. This includes application specific user preferences, for all applications, roles and permissions.

## Constituent Contact Management

An application that integrates external user profile information (from the user database) and operational information (history of interactions with the organization) in the context of constituent management. The application integrates loosely with other components on the client-side (e.g. ability to open a service request associated with a particular constituent).

## Enterprise Calendar

A standard events calendar application, but one that is ontology driven. While modeling of events and the information associated with them in an ontology is natural, the user-interface here is rather specific and will most likely necessitate custom programming. In addition, while events are a form of informational content, their dynamic nature requiree a separate, unified view independent of their associated departments, programs etc.

## Survey Creation and Conducting

In addition to standard survey functionality, in-place (contextual) feedback mechanisms are introduced pertaining to particular services or content delivered online.

## Idea Machine

The Idea Machine is a feedback application allowing users to submit ideas for improvement of the organization's operations. Ideas are business objects with a lifecycle similar to service requests in that they are handled ultimately by a specific department and moderated by a dedicated administrator, going through a set of states (statuses) with an ultimate outcome.

## Integration Services

The Integration Service is a set of loosely coupled components that act as connectors between the CiRM ecosystem and other enterprise systems whose functionality overlaps and/or complements CiRM. Since most existing systems have a CiRM oriented business function, opportunities for potential integration exist with nearly all software within the organization. For example, in addition to the legacy departmental CiRM solutions that will live in conjunction with the CiRM, online applications that performs certain services like payments, or applications for various services are good candidates for feature embedding as CiRM components.

The Integration Services should be entirely decoupled from the core CiRM components in the same way they will be decoupled from the external application being integrated. That is, no CiRM application should have any dependency on an integration service.

## Admin Component Suite

The Admin Component Suite is comprised of several administrative components that wrap the ontology based model in a more business user friendly interface. The components can be combined in various ways to create administrative dashboards for different user roles. A non-exhaustive list of those components is:

* Service Request Administration: managed service request types, workflows and forms.
* Content Manager Administration - manage content, semantic tagging, etc.
* Search Administration
* Survey Administration
* Idea Machine Administration
* Constituent Management Administration
* E-Campaign Management - email and reverse 311 notifications.

Other vertical administrative components are developed on an as-needed basis to offload administrative work from ontology savvy experts.