The Mitopia® Platform

June 2003

Below is a high level synopsis about what **Mitopia®** is, why it is unique, and how it represents an enormous strategic information advantage to commercial and government intelligence professionals. This document is not intended to be exhaustive; it is intended to delineate Mitopia® as an expansive, ontology-based, unifying architecture as opposed to a confined application. Mitopia® is an information architecture, or platform, designed with a radically new approach to the implementation of software applications and the usage of computer systems in general. It has been installed and running in production environments for over five years.

50,000 Foot View

It is common to think of a software system like Mitopia® as an application. It is not. It is an ontology-based and ontology-driven architecture, or operating environment, which is configured to *become* any number of applications through a discrete set of configuration components. This is a significant distinction. Software applications are most often designed from a set of very detailed specifications, built into prototypes, and then developed and deployed. The life cycle is typically measured in years. However, in a world where we face rapidly changing threats and highly mobile adversaries, we need to develop software systems that are equally, if not more, adaptable than the subject we are trying to study. That means systems must be able to be developed, modified, and deployed in a matter of days or weeks at the *most*. The only way to do this is to break from the traditional application mindset, and tackle the problem at the architectural level. Mitopia® has been in development for ten years to address that exact architectural problem. A very discrete set of configuration components can rapidly transform the architecture into a highly adaptable, targeted application in a matter or hours or days.

Evolving System versus Specified System Approach

Mitopia® defies most conventional notions of how computer software is written today. To fully appreciate it, one must reconsider the fundamental nature of the problem being solved, and break from the fixed mindset that there must be a top-down application-type solution. The thought that we can know *today* what problem we need to solve *tomorrow* is somewhat delusional. By the time a system is specified, and even put on the drawing board, the requirements have already changed - the world has changed. Change is the norm and needs to be part of the fundamental character of the system if it is to be of any use down the road. So we must think completely differently about solving the true problem.

The answer is to turn the equation on its head. Instead of designing systems from a deterministic, controlling, perspective, a software environment is needed where the application can *evolve,* where the individual user can define and redefine its functionality at any time. This sounds like science fiction but it is not. Mitopia® is an environment for supporting real time systems with all the facilities necessary to manage and fuse disparate data sources into a distributed, yet completely integrated, information repository. Layers of abstraction allow users to simply focus on what application functionality they need at any time. No longer do they have to make requests to hoards of database administrators or engineers and suffer long turn around times. Mitopia® handles it.

Consider a group of analysts in a workshop with any number of algorithms as their tools. Imagine a system connected to any and every data feed and database in the world – in any language. Through Mitopia's powerful, ontology-driven data model, all these sources can be fused, stored, and queried in any number of languages. Now, given the same problem, every analyst will approach it differently with different tools because every analyst has a different point of view. So who is right? The point is, you don't know. The analysts have to be able to develop their own models and see which works best as a solution. These models need to be continually changed, given changes in the external real world environment. This reinforces the argument that you cannot predefine system functionality the analyst will use. You have to allow them to *express* it to the system. *They* need to determine how the system should work rather than the system telling them how they must use it.

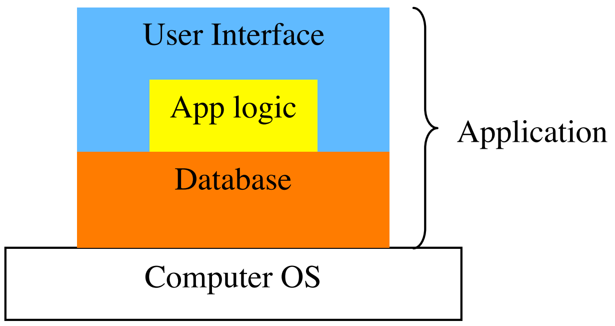
Data Flow versus Control Flow

The computing paradigm utilized is called *"data-flow,"* whereas virtually every other software system operating today relies on a limited "control flow" model. Control-flow involves the presumption that the program is in control of the data. This means the software application has to go out and inspect its inputs and perform operations according to some preordained set of instructions. This is how most software is written, and how most software engineers think. Of course, this is another example of human misperception that we can actually be in control of *anything* in the outside world. Look out your window. Can you write a program to describe how everything is connected and predict behavior from all the associated interactions? No. But that's how the real world works. So we need to model our software systems along at least a similar paradigm if we have any hope of successfully understanding events in the world in a real time manner. Data happens, and the occurrence of one event alters another. It is a chaotic system. Yet we desperately cling to our notion of trying to control it, to put order to it with our control-flow system mentalities. Ultimately, this will not work, no matter how hard you try. When you *invert* the equation to data-flow, the approach to system development completely changes. Now you have a model which properly mimics the subject you are studying: chaotic systems, i.e., the real world.

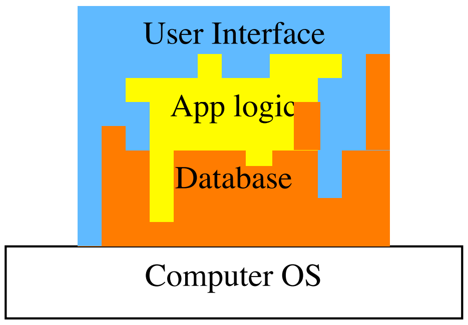
In data-flow systems, the *data* is "in control." This means application functionality is triggered as data flows into the system. Instead of a large, monolithic program structure, localized control-flow logic executes specific algorithmic functionality on the data depending on its type and attributes. We call these "widgets" (more on this later). This agent-based model represents an inversion of how complex systems are developed today. System-wide, the architectural substrate is more of a chaotic environment, allowing unexpected interactions to produce outputs that are not immediately obvious to the user. In this way, we are using machines for the purpose to which they are best suited: acting on massive amounts of information, and processing it real time, in more conceptual dimensions that a human analyst can even fathom. Through this approach, connections in data, or emerging trends can be detected and alerted to the human analyst. Then the analyst can do what they do well – synthesize the information and deliver it in a way that can become easily and rapidly actionable by the appropriate decision maker(s). Today's systems do little more than overload the analysts and force *them* to do the laborious, manual work of sifting through the information for nuggets of value. Mitopia® presents an opportunity to radically increase the productivity and efficiency of the human analyst, and assisting rapid response to quickly developing situations. National security, in particular, rests on the ability to provide this. Outdated systems and techniques will never get us ahead of a nimble and distributed adversary.

Architecture versus Application

The classic software application is basically made up of three main components: some persitent storage mechanism ("database"), the user interface ("UI"), and the application logic. Typically, a software team is broken into these generic areas, and they must closely coordinate each step of the way during project development. A change in one area can cause problems or require changes in the other area(s). This is the main reason why software systems are slow to adapt to change, and why one application is usually incompatible with another ("stovepipes"). An ideal software application might be modeled as below:

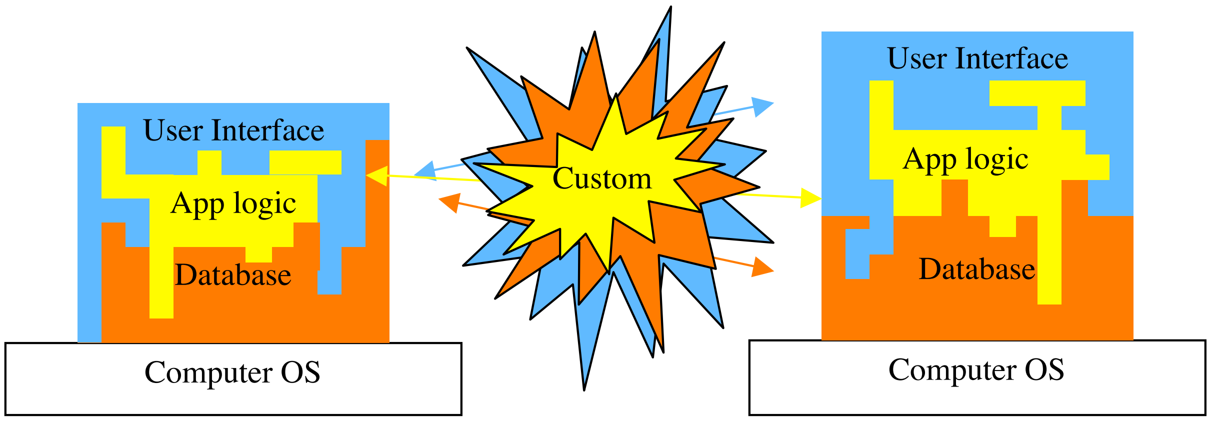


In theory, the underlying computer operating system provides all kinds of utilities such as file management, scheduling, etc., for use by an application. However, in reality, the application (especially a complex intelligence application) most often requires its own set of specialized utilities and functions. This represents an enormous amount of software code that is custom only to that application, and becomes increasingly difficult to manage as the complexity and size of the application increases. The picture would look more like the following:



There is an enormous amount of custom code associated with getting data in and out of storage (relational databases), and presenting it through the user interface. Any time a request comes for some functional change, all these areas must be consulted and likely modified. The result, inevitably, is a tangled mess of code that is difficult to decipher by anyone other than the team who created it.

Now consider attempting to get two different applications to work together. This is what so many have tried to do in integrating multiple commercial-off-the-shelf ("COTS") products. It would look something like this:



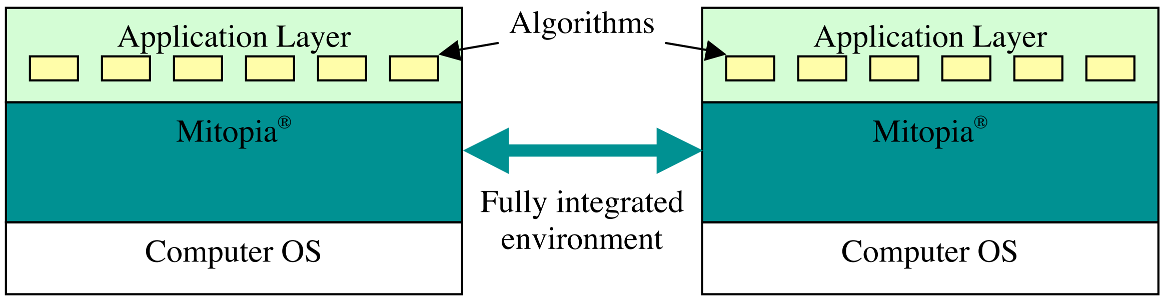
COTS applications are typically designed and produced by different companies with different philosophies, protocols, programming languages, etc. In addition, each application, by nature, wants to be in control. They are not designed to be compatible with other applications. Taking this patchwork approach is a Frankenstein in the making. It is doomed to failure. An architectural approach must be taken.

Mitopia® Operating Environment

A conventional operating system ("OS") provides a variety of services for the individual computer. But when we are trying to build massive, highly distributed, multimedia, multilingual intelligence applications, a new layer of abstraction is necessary. Mitopia® provides this layer, and is intended to support all such functions. In fact, it takes away control from much of the operating system functions such as scheduling and memory management. But it uses some of the OS tools where it is more convenient. At the same time, Mitopia® provides all the services required of an application, including the generation of all persistent storage and the presentation of the user interface *automatically*. This takes out a *large* portion of the custom code overhead associated with an application.

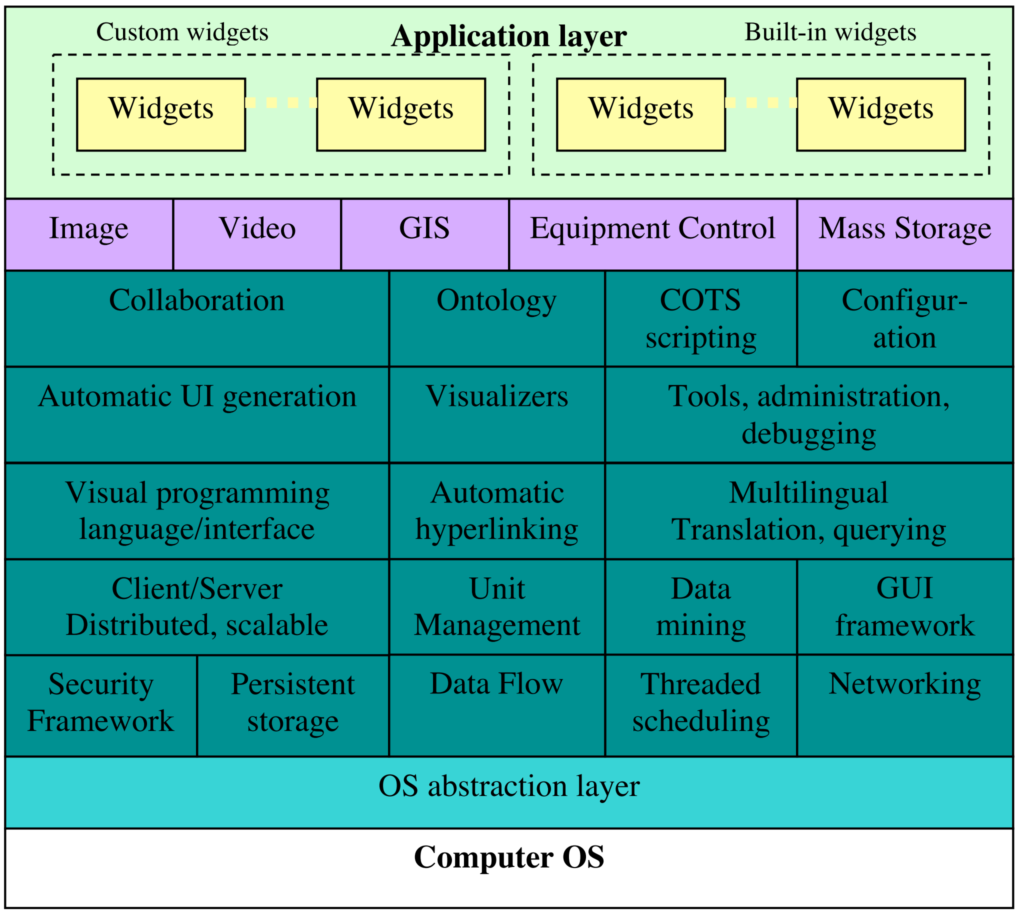
The value in each COTS application usually comes down to a very limited set of core algorithms. These are the true value buried within the application, and companies have built entire businesses (and custom application code) around these few algorithms. In other words, all that is needed from the COTS application is the core algorithmic component(s).

With Mitopia® the picture looks like the following:



Algorithms (one type of "widget") from a host of different sources can be mixed and matched to produce functionality, as desired, against any of the data sources connected through the environment.

A Closer Look within Mitopia®:



What is a "Widget?"

"Widget" is the generic term for any functional, logical component, which helps build up the particular application functionality. A widget could represent an algorithm, or it could represent a specific process, or a particular visual representation. Widgets are what users connect to produce application functionality. Widgets are actually collections of low-level, executable instructions. Widgets represent only the most basic level of functionality within Mitopia® and cannot execute by themselves. Widgets either can be *atomic* (containing compiled code that cannot be further modified or examined by the normal user), or *compound* (containing a *series* of other atomic and/or compound widgets, possibly nested, which are interconnected via Mitopia® engine data flows, and whose behavior potentially can be examined and/or modified by normal users). Widgets are combined and edited using a visual application-programming interface called "Widget Editing Mode," or WEM. Atomic widgets are created and compiled solely by software engineers.

Subsystems

In the diagram above you will see a number of subsystems identified in purple. These are specific to certain types of data in the environment.

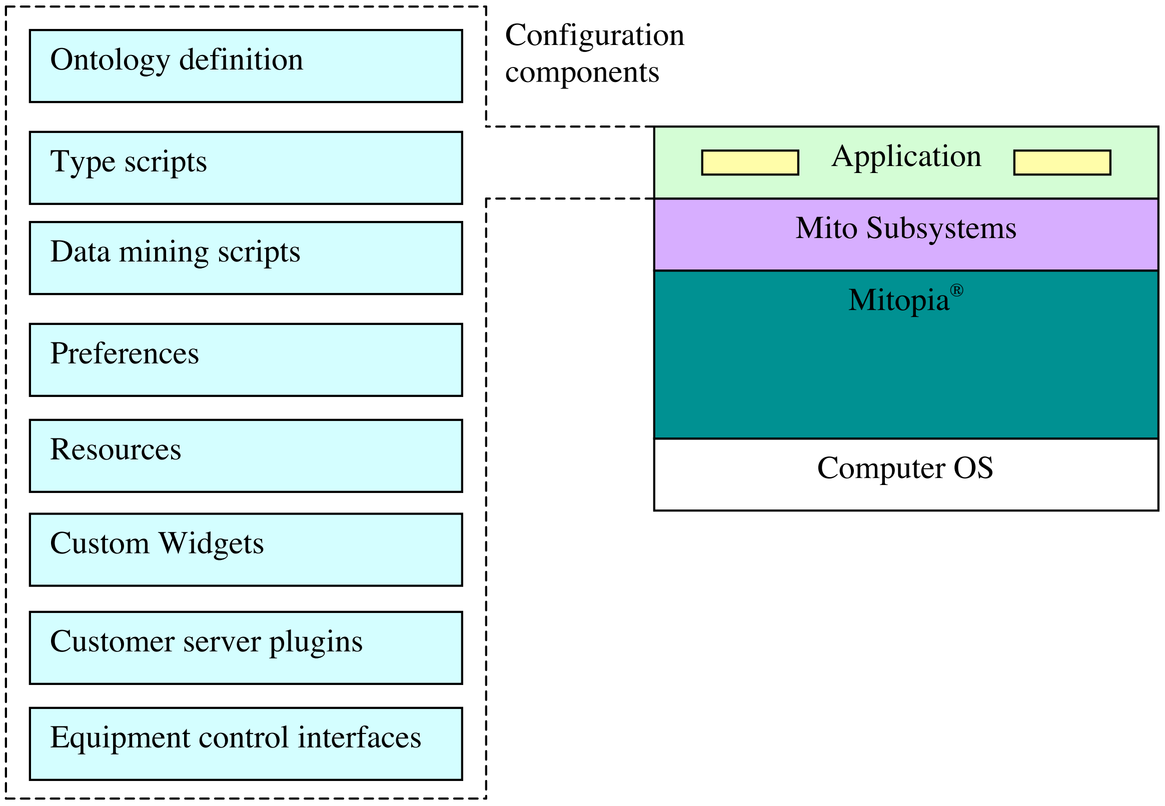
Video and Image Subsystems require significantly different support characteristics and tools than just textual information. A complete capability exists for handling these data types in a distributed manner.

Equipment Control and Mass Storage Subsystems are necessary for managing all aspects of information ingestion at the source and storing it away for retrieval later. This is especially important for high bandwidth information such as images and video.

The **Geographic Information Subsystem ("GIS")** is one of the more unique aspects of the Mitopia® environment. Instead of trying to fuse traditional GIS capabilities at the application level, Mitopia® handles everything at the *data* level. Through the mechanism of the ontology, every piece of data entering the system can be associated with some spatial location. The key is that these elements are interactive so after projecting them onto geospatial coordinates, an analyst can manipulate them further, perhaps to pear down a set, and then project them back into some other form for further visual analysis and distinction of patterns. GIS in Mitopia® is just a special projection of data – along the "where" axis. Time (the "when" axis) and well as any number of other "axes" can be created to analyze information in meaningful ways. This must all be interactive at the data level so information can be pruned and sorted in a number of contexts. This is the most effective way to find the proverbial needle(s) in the enormous haystack of information.

Configuration

Mitopia® is a highly leveraged environment. Layers upon layers of abstraction have been developed such that only a few components need to be configured to set up and launch an entire application on any scale.



Description of Configuration Components

System Ontology

The Ontology is the formalization through which the classification and categorization of data is defined. The Ontology defines what you can extract from incoming data. It determines what gets stored where/how, and it determines what gets displayed throughout the application. All data flowing into the system and use of the system is organized according to the Ontology.

Type Scripts

Mitopia® type manager script functions customize the behavior of various types and type fields defined in the Ontology in response to a variety of predefined logical actions. Through this mechanism, it is possible to alter most aspects of application querying, server communications, default values, and data inter-linking behaviors. The Mitopia® environment uses the mechanism of type scripts to avoid any explicit dependency on application-specific functionality or behaviors; instead, all customizable Mitopia® behaviors relating to its interaction with other subsystems is mediated through the combination of the Ontology and the type scripts.

Data Mining Scripts (MitoMine™)

MitoMine™ is MitoSystems' powerful data extraction technology. It allows complete user control over the extraction of information from unstructured, loosely structured, and diverse sources into a single, normalized, form as defined by the system Ontology. As new sources become available, the Ontology changes, or existing sources change format, MitoMine™ scripts can be altered to allow the new/changed source to be ingested.

Preferences

Systems based on the Mitopia® architecture can be rapidly re-configured and customized in a variety of ways by use of Mitopia's preference mechanism. Preferences can be used to configure servers and server clusters to machines, change user interface options, or select from any number of alternative approaches supported by any software running under the environment. The advantage of this preference-based approach is the ability to rapidly and easily alter system behavior and options from the application itself without the need to alter source code.

Resources

In order to customize the UI appearance of various windows, a large number of resources are required. These resources can be changed in order to customize the appearance and behavior of a given application via the UI framework. In this context, resource includes UI configuration items such as XML and various image files.

Custom Widgets

Widgets interact through the Mitopia® environment, primarily via data-flow, and can be assembled and combined in order to create increasingly complex functionality and behaviors. Any specific application is likely to need one or more custom widgets to be implemented in order to provide some aspect of the functionality unique to that application. Examples of widgets might be custom analytical tools, domain specific algorithms, custom user interface components, etc. The Mitopia® environment provides a rich suite of application programming interfaces ("API") that can be used by such custom widgets in order to rapidly create new functionality that is fully integrated with the environment.

Custom Server Plug-ins

Whenever a new multimedia data type is added to the system, or a new source is connected that requires custom code in order to interface with it or control it, custom server plug-in modules must be developed. Not all systems will require this since MitoSystems provides many standard servers and the necessary plug-ins as part of the core software. The Mitopia® server architecture and the corresponding API calls allow for the rapid and robust creation of custom servers simply by registering a relatively small number of standardized logical callback functions.

Equipment Control Interfaces

In systems involving the integration and control of external equipment as part of system operation (a feature fully supported by the architecture), it will almost certainly be necessary to develop custom equipment drivers, or to obtain equipment control information compatible with an existing system driver. Development of equipment drivers can be quite complex, but is made easier by the rich support provided within Mitopia® for this purpose.

Summary

Designing and delivering a large software application can take years. This is not acceptable when dealing with a rapidly changing environment or threat. Attempting to glue and paste together COTS applications will only create a complicated, inconsistent, patchwork mess. A bottom-up, architectural, approach is required. An architectural framework must be utilized to provide all the facilities necessary for launching highly sophisticated intelligence applications, enable incorporation of algorithmic components from commercial or government technologies, and provide a seamless, flexible, and rapidly adaptable system for use by anyone from analyst to decision maker. MitoSystems has been singularly focused on this mission, with Mitopia® in development for over ten years. Operational systems have been successfully deployed and proven in production environments for over five years. What's in *your* system?

Mitopia®: The Definitive Intelligence Platform.

For more information, please contact Ted Whetstone, Business Development, at (310) 581-3600 ext. 228 or tedw@mitosystems.com