C3.ai

A New Technology Stack

Digital transformation requires a new software technology stack and a new approach to developing enterprise AI applications.

Introduction

Over the last four decades, the information technology industry has grown from about $50 billion globally in 1980 to more than $4 trillion. During this time, the IT industry has experienced the transition from mainframe computing to minicomputers, to personal computing, to internet computing, and to handheld computing. The software industry has transitioned from custom applications based on mainframe standards such as MVS, VSAM, and ISAM, to applications developed on a relational database foundation, to enterprise application software, to SaaS and mobile apps, and now to the AI-enabled enterprise. The internet and the iPhone changed everything.

Each of these transitions represented a replacement market for its predecessor. Each delivered dramatic benefits in productivity. Each offered organizations the opportunity to gain sustainable competitive advantage. Companies that failed to take advantage of each new generation of technology ceased to be competitive. Today it is unimaginable that a major global corporation would try to close its books without an enterprise resource planning system or run its business solely on mainframe computers.

The IT industry is now undergoing another major transition. A new generation of 21st century technologies – including elastic cloud computing, the internet of things, and artificial intelligence – is driving digital transformation across industry, commerce, and government globally. Digital transformation presents a number of unique requirements that create the need for an entirely new software technology stack. The requirements are daunting.

This paper defines the requirements of the new digital transformation software stack. It explains why the current approach to developing AI and IoT enterprise software – i.e., using structured programming to build applications by assembling and integrating various open source components and cloud services – is slow, costly, and ineffective.

Finally, this paper describes how the C3 AI Suite with its unique model-driven architecture fully addresses the requirements for the digital transformation software stack, providing a low-code/no-code AI and IoT platform that accelerates software development by a factor of 40 or more, reduces cost and risk, and delivers future-proof applications.

Requirements of the New Technology Stack

To develop an effective enterprise AI or IoT application, it is necessary to aggregate data from across thousands of enterprise information systems, suppliers, distributors, markets, products in customer use, and sensor networks, in order to provide a near-real-time view of the extended enterprise.

Today's data velocities are dramatic, requiring the ability to ingest and aggregate data from hundreds of millions of endpoints at very high frequency, sometimes exceeding 1,000Hz cycles. The data need to be processed at the rate they arrive, in a highly secure and resilient system that addresses persistence, event processing, machine learning, and visualization. This requires massively horizontally scalable elastic distributed processing capability offered only by modern cloud platforms and supercomputer systems.

The resultant data persistence requirements are staggering. These data sets rapidly aggregate into hundreds of petabytes, even exabytes. Each data type needs to be stored in an appropriate database capable of handling these volumes at high frequency. Relational databases, key-value stores, graph databases, distributed file systems, and blobs are all necessary, requiring the data to be organized and linked across these divergent technologies.

Reference AI Software Platform

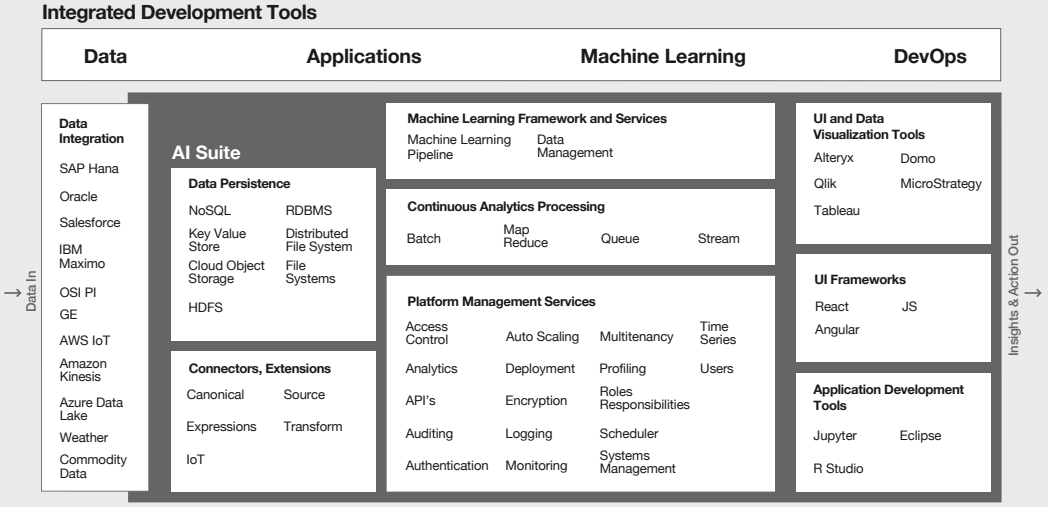
The problems that have to be addressed to enable today's AI and IoT applications are nontrivial. Massively parallel elastic computing and storage capacity are prerequisite. These services are provided today at increasingly low cost by Microsoft Azure, AWS, and others. The elastic cloud is a major breakthrough that has dramatically transformed modern computing. In addition to the cloud, multiple data services are necessary to develop, provision, and operate AI and IoT applications.

Figure 1 depicts the array of capabilities and services necessary for the domain of AI and IoT applications. Each of these utilities represents a development problem on the order of magnitude of a relatively simple enterprise software application such as CRM. This is not a trivial problem. Let's take a look at some of these requirements.

Figure 1

Reference Architecture for AI Suite

The successful development of AI and IoT applications requires a complete suite of tools and services that are fully integrated and designed to work together.



Data Integration: This problem has haunted the computing industry for decades. Prerequisite to machine learning and AI at industrial scale is the availability of a unified, federated image of all the data contained in the multitude of (1) enterprise information systems – ERP, CRM, SCADA, HR, MRP – typically thousands of systems in each large enterprise; (2) sensor IoT networks – SIM chips, smart meters, programmable logic arrays, machine telemetry, bioinformatics; and (3) relevant extraprise data – weather, terrain, satellite imagery, social media, biometrics, trade data, pricing, market data, etc.

Data Persistence: The data aggregated and processed in these systems includes every type of structured and unstructured data imaginable. Personally identifiable information, census data, images, text, video, telemetry, voice, network topologies. There is no “one size fits all” database that is optimized for all these data types. This results in the need for a multiplicity of database technologies including but not limited to relational, NoSQL, key-value stores, distributed file systems, graph databases, and blobs.

Platform Services: A myriad of sophisticated platform services are necessary for any enterprise AI or IoT application. Examples include access control, data encryption in motion, encryption at rest, ETL, queuing, pipeline management, autoscaling, multitenancy, authentication, authorization, cybersecurity, time-series services, normalization, data privacy, GDPR privacy compliance, NERC-CIP compliance, and SOC2 compliance.

Analytics Processing: The volumes and velocity of data acquisition in such systems are blinding and the types of data and analytics requirements are highly divergent, requiring a range of analytics processing services. These include continuous analytics processing, MapReduce, batch processing, stream processing, and recursive processing.

Machine Learning Services: The whole point of these systems is to enable data scientists to develop and deploy machine learning models. There is a range of tools necessary to enable that, including Jupyter Notebooks, Python, DIGITS, R, and Scala. Increasingly important is an extensible curation of machine learning libraries such as TensorFlow, Caffe, Torch, Amazon Machine Learning, and AzureML. An effective AI and IoT platform needs to support them all.

Data Visualization Tools: Any viable AI architecture needs to enable a rich and varied set of data visualization tools including Excel, Tableau, Qlik, Spotfire, Oracle BI, Business Objects, Domo, Alteryx, and others.

Developer Tools and UI Frameworks: An organization’s IT development and data science teams each have adopted and become comfortable with a set of application development frameworks and user interface (UI) development tools. An AI and IoT platform must support all of these tools – including, for example, the Eclipse IDE, VI, Visual Studio, React, Angular, R Studio, and Jupyter – or it will be rejected as unusable by the IT development teams.

Open, Extensible, Future-Proof: The current pace of software and algorithm innovation is blinding. The techniques used today will be obsolete in 5 to 10 years. An AI and IoT platform architecture must therefore provide the capability to replace any components with their next-generation improvements. Moreover, the platform must enable the incorporation of any new open source or proprietary software innovations without adversely affecting the functionality or performance of an organization’s existing applications. This is a level-zero requirement.

To meet this extensive set of requirements, C3.ai has spent the last decade and invested more than $500 million in developing and enhancing the C3 AI Suite. The C3 AI Suite has been refined, tested, and proven in the most demanding industries and production environments – electric utilities, manufacturing, oil and gas, and defense – comprising petabyte-scale datasets from thousands of vastly disparate source systems, massive volumes of high-frequency time series data from millions of devices, and hundreds of thousands of machine learning models.

Awash in “AI Platforms”

Industry analysts estimate that organizations will invest more than $250 billion annually in digital transformation software by 2025. According to McKinsey, companies will generate more than $20 trillion annually in added value from the use of these new technologies. This is the fastest-growing enterprise software market in history and represents an entire replacement market for enterprise application software.

Today the market is awash in open source “AI Platforms” that purport to be solutions sufficient to design, develop, provision, and operate enterprise AI and IoT applications. In this era of AI hype, there are literally hundreds of these in the market – and the number increases every day – that present themselves as comprehensive “AI Platforms.”

Figure 2

A Sea of “AI Platforms”

The market is awash with hundreds of open source components that purport to be an “AI platform.” Each component can provide value, but none provides a complete platform by itself.

Examples include Cassandra, Cloudera, DataStax, Databricks, AWS IoT, and Hadoop. AWS, Azure, IBM, and Google each offer an elastic cloud computing platform. In addition, each offers an increasingly innovative library of microservices that can be used for data aggregation, ETL, queuing, data streaming, MapReduce, continuous analytics processing, machine learning services, data visualization, etc.

They all appear to do the same thing and they all appear to provide a complete AI platform. While many of these products are useful, the simple fact is that none offers the scope of utility necessary and sufficient to develop and operate an enterprise AI or IoT application.

Consider Cassandra, for example. It is a key-value data store, a special-purpose database that is particularly useful for storing and retrieving longitudinal data, like telemetry. For that purpose, it is an effective product. But that functionality represents perhaps one percent of the required solution. Likewise, HDFS is a distributed file system, useful for storing unstructured data. TensorFlow, a set of math libraries published by Google, is useful in enabling certain types of machine learning models. Databricks enables data virtualization, allowing data scientists or application developers to manipulate very large data sets across a cluster of computers. AWS IoT is a utility for gathering data from machine-readable IoT sensors. The point is: these utilities are all useful, but none is sufficient by itself. Each addresses only a small part of the problem required to develop and deploy an AI or IoT application.

Moreover, these utilities are written in different languages, with different computational models and frequently incompatible data structures, developed by programmers of varying levels of experience, training, and professionalism. They were not designed to work together. Few, if any, were written to commercial programming standards. Most have not proven commercially viable and the source code has been contributed to the open source community. The open source community is a kind of superstore in the cloud with a growing collection of hundreds of computer source code programs available for anyone to download, modify at will, and use at no cost.

“Do It Yourself” AI?

Software innovation cycles follow a typical pattern. Early in the cycle, companies often take a “do it yourself” approach and try building the new technology themselves. In the 1980s, for example, when Oracle first introduced relational database management system (RDBMS) software to the market, interest was high. RDBMS technology offered dramatic cost economies and productivity gains in application development and maintenance. It proved an enabling technology for the next generation of enterprise applications that followed, including material requirements planning (MRP), enterprise resource planning (ERP), customer relationship management (CRM), manufacturing automation, and others.

The early competitors in the RDBMS market included Oracle, IBM (DB2), Relational Technology (Ingres), and Sperry (Mapper). But the primary competitor to Oracle was not any of these companies. It was in many cases the CIO, who attempted to build the organization’s own RDBMS with IT personnel, offshore personnel, or the help of a systems integrator. None of those efforts succeeded. Eventually, the CIO was replaced and the organization installed a commercial RDBMS.

When enterprise applications including ERP and CRM were introduced to the market in the 1990s, the primary competitors included Oracle, SAP, and Siebel Systems. But in the early years of that innovation cycle, many CIOs attempted to develop these complex enterprise applications internally. Hundreds of person-years and hundreds of millions of dollars were spent on those projects. A few years later, a new CIO would install a working commercial system.

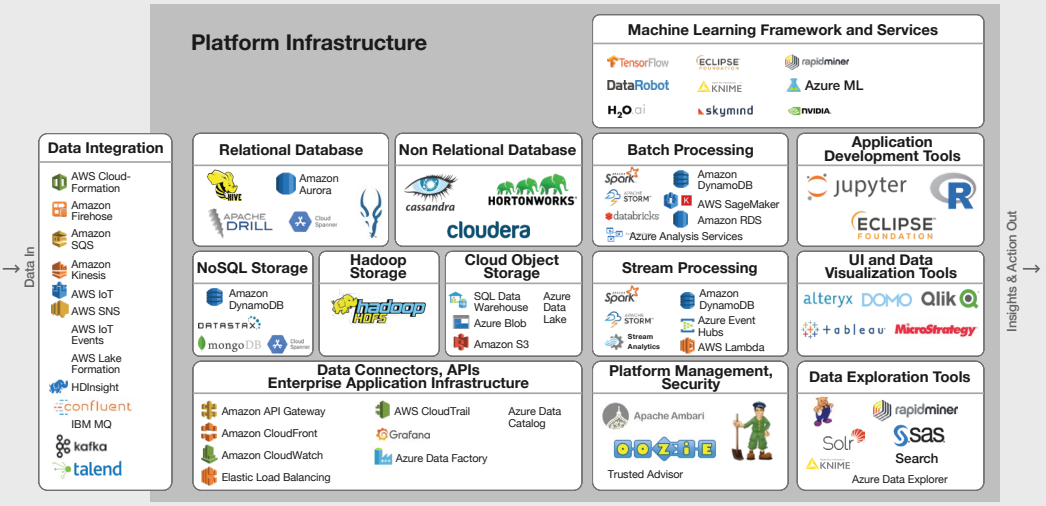
Some of the most technologically astute companies – including Hewlett-Packard, IBM, Microsoft, and Compaq – repeatedly failed at internally developed CRM projects. All ultimately became successful Siebel Systems CRM customers.

Just as happened with the introduction of RDBMS, ERP, and CRM software in prior innovation cycles, the initial reaction of many IT organizations is to try to internally develop a general-purpose AI and IoT platform, using open source software with a combination of microservices from cloud providers like AWS and Google. The process starts by taking some subset of the myriad of proprietary and open source solutions and organizing them into the reference platform architecture depicted in Figure 3.

Figure 3

The AI Software Stack

The “build it yourself” approach requires stitching together dozens of disparate open source components from different developers with different APIs, different code bases, and different levels of maturity and support.



The next step is to assemble hundreds to thousands of programmers, frequently distributed around the world, using structured programming and application programming interfaces (APIs) to attempt to stitch these various programs, data sources, sensors, machine learning models, development tools, and user interface paradigms together into a unified, functional, seamless whole that will enable the organization to excel at designing, developing, provisioning, and deploying numerous enterprise scale AI and IoT applications.

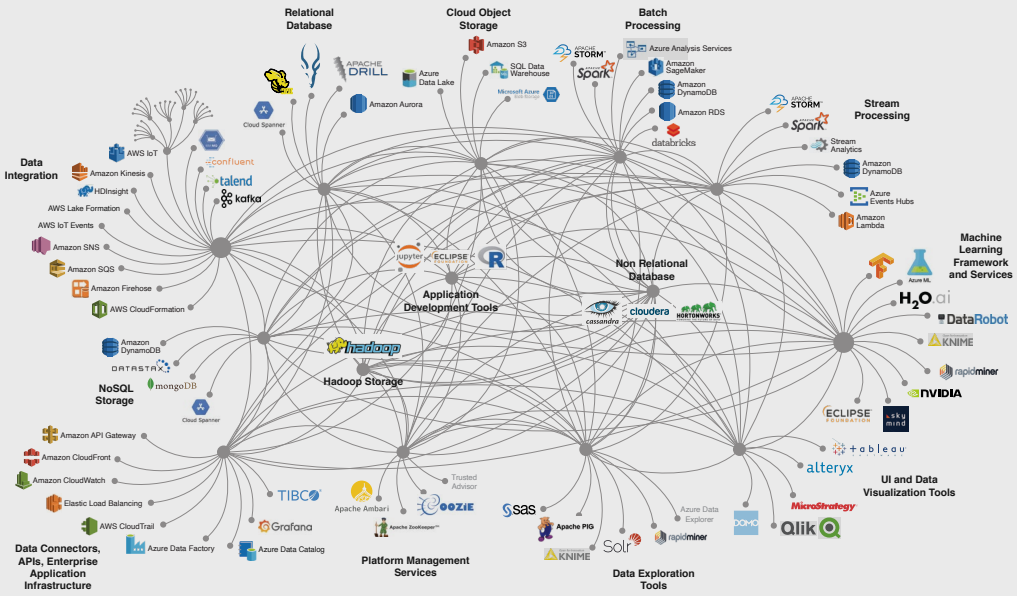
The complexity of such a system is two orders of magnitude greater than developing a CRM or ERP system. Many have attempted to build such a system, and none have succeeded. The classic case study is GE Digital that expended eight years, 3,000 programmers, and $7 billion trying to succeed at this task. The end result of that effort included the collapse of that division and the termination of the CEO, and it contributed to the dissolution of one of the world's iconic companies.

Were someone to succeed at such an effort, the resultant software stack would look something like Figure 4.

Figure 4

AI Software Cluster

The "build it yourself" approach requires numerous integrations of underlying components that were not designed to work together, resulting in a degree of complexity that overwhelms even the best development teams.



There are a number of problems with this approach:

1. Complexity

Using structured programming, the number of software API connections that one needs to establish, harden, test, and verify for a complex system can approach the order of 10^{13}. The developers of the system need to individually and collectively grasp that level of complexity to get it to work. The number of programmers capable of dealing with that level of complexity is quite small.

Aside from the platform developers, the application developers and data scientists also need to understand the complexity of the architecture and all the underlying data and process dependencies in order to develop any application. The level of complexity inherent in these efforts is sufficiently great to assure project failure.

2. Brittleness

Spaghetti-code applications of this nature are highly dependent upon each and every component working properly. If one developer introduces a bug into any one of the open source components, all applications developed with that platform may cease to function.

3. Future Proof

As new libraries, faster databases, and new machine learning techniques become available, those new utilities need to be available within the platform. Consequently, every application that was built on the platform will likely need to be reprogrammed in order to function correctly. This may take months to years.

4. Data Integration

An integrated, federated common object data model is absolutely necessary for this application domain. Using this type of structured programming API-driven architecture will require hundreds of person-years to develop an integrated data model for any large corporation. This is the primary reason why tens to hundreds of millions of dollars get spent, and several years later, no applications are deployed. The Fortune 500 is littered with such disaster stories.

The Gordian Knot of Structured Programming

Structured programming is a technique introduced in the mid-1960s to simplify code development, testing, and maintenance. Prior to structured programming, software was written in large monolithic tomes replete with APIs and “go-to” statements. The resultant product might consist of millions of lines of code with thousands of such APIs and go-to statements that were difficult to develop, understand, debug, and maintain.

The essential idea of structured programming was to break the code into a relatively simple “main routine” and then use something called an application programming interface (API) to call subroutines that were designed to be modular and reusable. Example subroutines might provide services like complete a ballistics calculation, or a fast Fourier transform, a linear regression, an average, a sum, or a mean. Structured programming remains the state of the art for many applications today, and has dramatically simplified the process of developing and maintaining computer code.

While this technique is appropriate for many classes of applications, it breaks down with the complexity and scale of the requirements for a modern AI or IoT application, resulting in a Gordian knot depicted in Figure 4.

Cloud Vendor Tools

An alternative to the open source cluster is to attempt to assemble the various services and microservices offered by the cloud providers into a working seamless and cohesive enterprise AI and IoT platform. As depicted in Figure 5, leading vendors like AWS are developing increasingly useful services and microservices that in many cases replicate the functionality of the open source providers and in many cases provide new and unique functionality. The advantage of this approach over open source is that these products are developed, tested, and quality assured by highly professional enterprise engineering organizations. In addition, these services were generally designed and developed with the specific purpose that they would work together and interact in a common system. The same points hold true for Google, Azure, and IBM.

Figure 5

Cloud Vendor Tools—AWS

Public cloud platforms like AWS, Azure, and Google Cloud offer an increasing number of tools and microservices, but stitching them together to build enterprise-class AI and IoT applications is exceedingly complex and costly.

The problem with this approach is that because these systems lack a model-driven architecture like that of the C3 AI Suite, described in the following section, programmers still need to employ structured programming to stitch together the various services. This results in the same type of complexity previously described – many lines of spaghetti code and numerous interdependencies that create brittle applications that are difficult and costly to maintain.

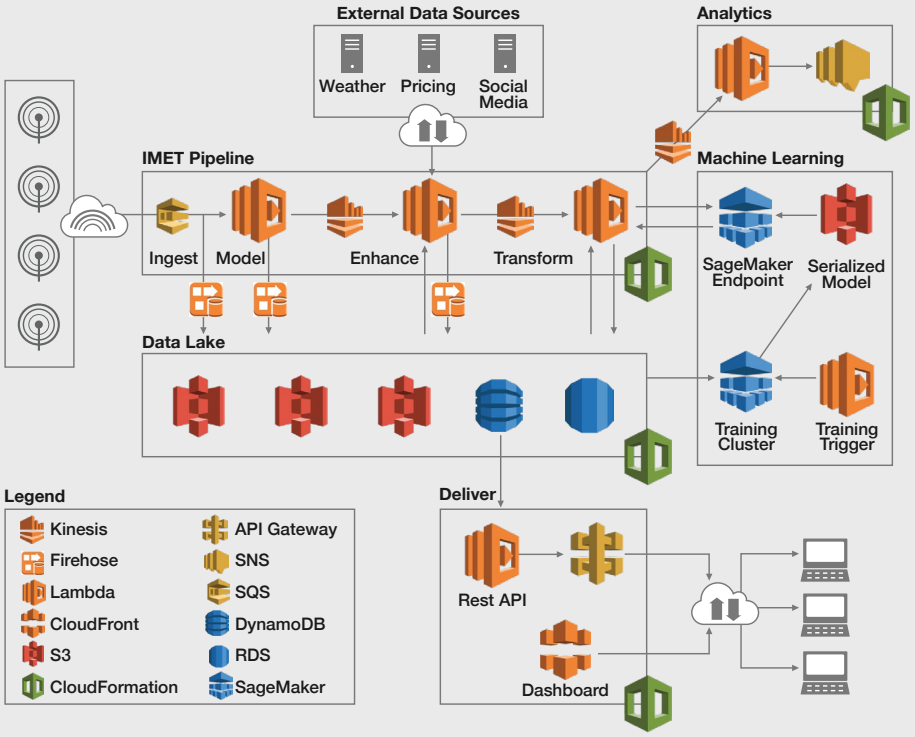
The difference between using structured programming with cloud vendor services and using the model-driven architecture of the C3 AI Suite is dramatic. To demonstrate this stark difference, C3.ai commissioned a third-party consultancy to develop an AI predictive maintenance application designed to run on the AWS cloud platform. The consultancy – a Premier AWS Consulting Partner, with significant experience developing enterprise applications on AWS for Fortune 2000 customers – was asked to develop the application using two different approaches: the C3 AI Suite and structured programming.

Using structured programming and AWS services, the reference architecture for this relatively simple predictive maintenance application is shown in Figure 6. The time to develop and deploy this application was 200 person-days at a cost in 2019 dollars of $600,000. The effort required writing 83,000 lines of code that must be maintained over the life of the application. The resulting application runs only on AWS. To run this application on Google or Azure, it would have to be completely rebuilt for each of those platforms at a similar cost, time, and coding effort.

Figure 6

Architecture to Build a Basic Predictive Maintenance Application on AWS

Building even a simple AI predictive maintenance application using microservices of a public cloud (AWS in this example) and a structured programming approach takes 40 times the work effort of using a model-driven architecture.



By contrast, using the C3 AI Suite with its modern model-driven architecture, the same application, employing the same AWS services, was developed and tested in 5 person-days at a cost of approximately $2,000. Only 1,450 lines of code were generated, dramatically decreasing the lifetime cost of maintenance. Moreover, the application will run on any cloud platform without modification, eliminating any additional effort and cost of refactoring the application if moving it to a different cloud vendor.

C3 AI Suite: Model-Driven Architecture

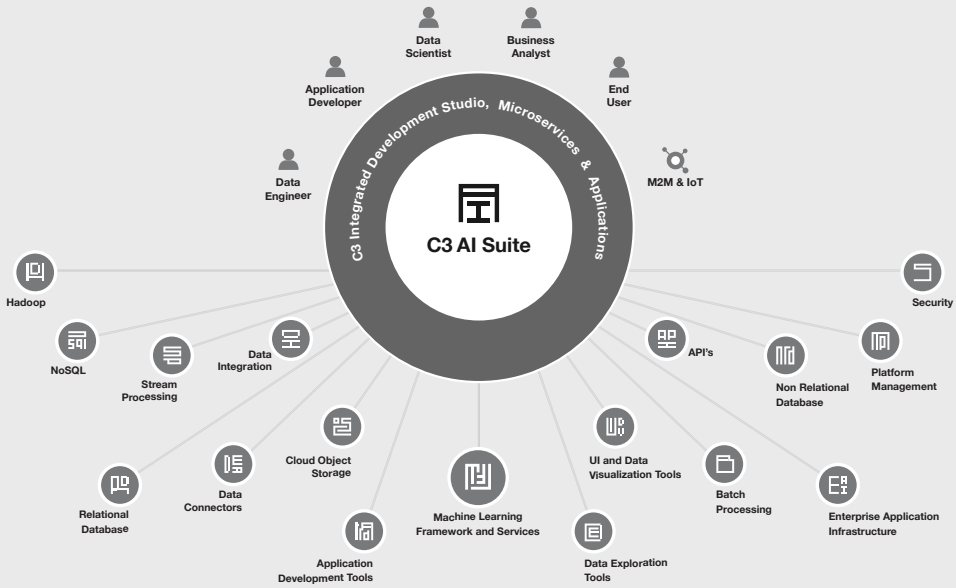
The notion of a model-driven architecture was developed at the beginning of the 21st century in response to the growing complexity of enterprise application development requirements. Model-driven architecture provides the knife to cut the Gordian knot of structured programming for highly complex problems. The C3 AI Suite is designed and built with a model-driven architecture.

Central to a model-driven architecture is the concept of a "model" that serves as an abstraction layer to simplify the programming problem. Using models, the programmer or application developer does not have to be concerned with all the data types, data interconnections, and processes that act on the data associated with any given entity, e.g., customer, tractor, doctor, or fuel type. He or she simply needs to address the model for any given entity – e.g., customer – and all the underlying data, data interrelationships, pointers, APIs, associations, connections, and processes associated with or used to manipulate those data are abstracted in the model itself. This reduces, from an order of 10^{13} to 10^{3}, the number of elements, processes, and connections of which the programmer or application developer needs to be aware, making the intractable now quite tractable.

Figure 7

Model-Driven AI Architecture

C3 AI Suite's model-driven architecture provides an abstraction layer that vastly simplifies and accelerates the development and deployment of AI and IoT applications.



Using the C3 AI Suite and its model-driven architecture, anything can be represented as a model – even, for example, applications, including databases, natural language processing engines, and image recognition systems. Models also support a concept called inheritance. An AI application built with the C3 Suite might include a model called relational database, that in turn serves as a placeholder that might incorporate any relational database system like Oracle, Postgres, Aurora, Spanner, or SQL Server. A key-value store model might contain Cassandra, HBase, Cosmos DB, or DynamoDB.

C3.ai Reduces Complexity, Simplifies Development

With its model-driven architecture, the C3 AI Suite provides an abstraction layer and semantics to represent the application. This frees the programmer from having to worry about data mapping, API syntax, and the mechanics of the myriad of computational processes like ETL, queuing, pipeline management, encryption, etc.

By reducing the number of entities, objects, and processes the developer needs to understand from an order of 10^{13} to 10^{3}, and by freeing the developer from wrestling with all this minutiae, the model-driven architecture of the C3 AI Suite decreases the cost and complexity of designing, developing, testing, provisioning, maintaining, and operating an application by as much as 100 times or more.

The optimal design for an object model to address AI and IoT applications uses abstract models as placeholders to which a programmer can link an appropriate application. The relational database model might link to Postgres. A report writer model might link to MicroStrategy. A data visualization model might link to Tableau. And so on. A powerful feature of a model-driven architecture is that as new open source or proprietary solutions become available, the object model library can simply be extended to incorporate that new feature.

Another important capability of the C3 AI Suite enabled by its model-driven architecture is that the applications developed on the platform are entirely future-proofed. Suppose, for example, that an organization developed all its applications initially using Oracle as the relational database and then later decided to switch to an alternate RDBMS. The only modification required is to change the link in the RDBMS meta-model to point to the new RDBMS. All the applications deployed previously using Oracle as the RDBMS will continue to run without modification after that replacement. This enables organizations to immediately and easily take advantage of new and improved product offerings as they become available.

Platform Independence: Multi-Cloud and Polyglot Cloud Deployment

The rate of cloud computing adoption in recent years has been dramatic and continues to accelerate. As recently as 2011, the message delivered by CEOs and corporate leadership worldwide was clear: “Our data will never reside in the public cloud.” The message today is equally clear: “We have a cloud-first strategy. All new applications are being deployed in the cloud. Existing applications will be migrating to the cloud. But understand, we have a multi-cloud strategy.”

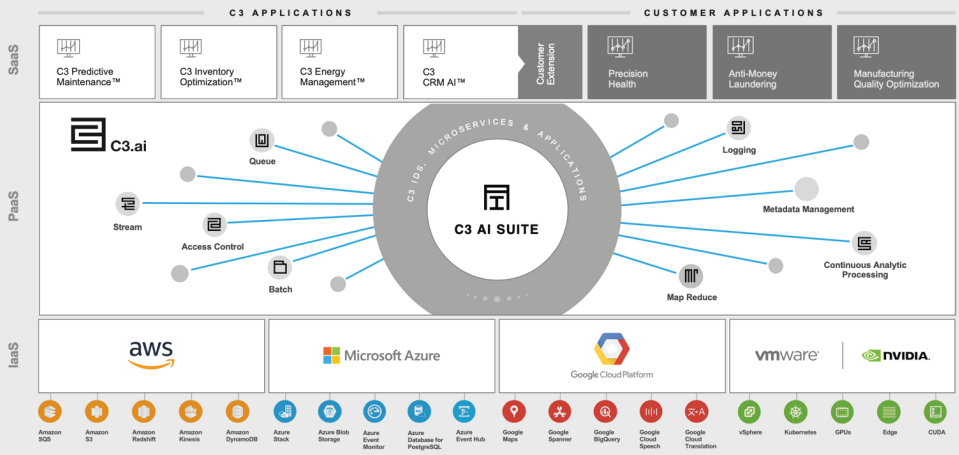
This 180-degree turn at global scale in the span of a few years is remarkable. But while corporate leaders are eagerly embracing the cloud, they are also very concerned about cloud vendor lock-in. They want to be able to continually negotiate. They want to deploy different applications in clouds from different vendors, and they want to be free to move applications from one cloud vendor to another.

Multi-cloud deployment is therefore an additional requirement of a modern model-driven software platform that is fully supported by the C3 AI Suite. Applications developed with the C3 AI Suite can run without modification on any cloud and on bare metal behind the firewall in a hybrid cloud environment.

Figure 8

Multi-Cloud Deployment

C3 AI Suite’s model-driven architecture enables organizations to deploy applications on multiple public cloud platforms as well as on bare metal behind the firewall in a private cloud or data center.



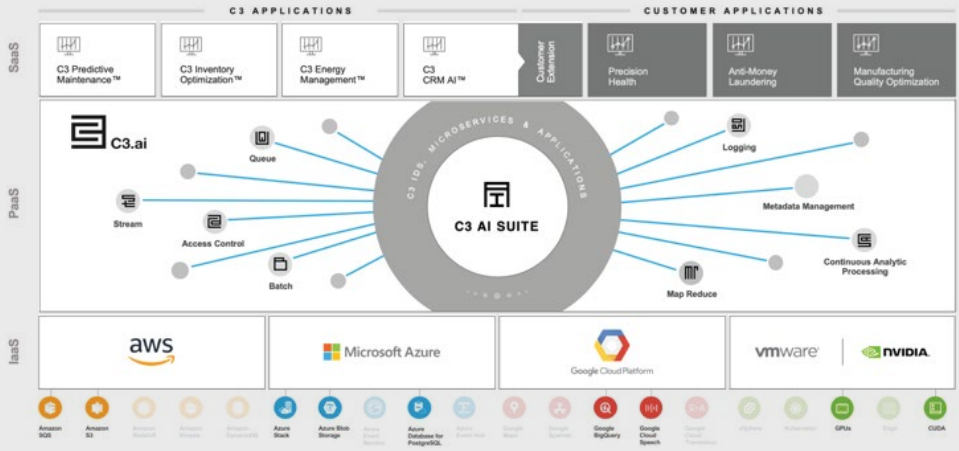
A final requirement for the new AI technology stack – that the C3 AI Suite fully delivers – is polyglot cloud deployment capability: the ability to mix various services from multiple cloud providers and to easily swap and replace those services. The cloud vendors provide the market a great service by enabling instant access to virtually unlimited horizontally scalable computing capacity and effectively infinite storage capacity at exceptionally low cost. As the cloud vendors aggressively compete with one another on price, the cost of cloud computing and storage is approaching zero.

A second important service cloud vendors provide is rapid innovation of microservices. Microservices like TensorFlow from Google accelerate machine learning. Amazon Forecast facilitates deep learning for time-series data. Azure Stream Analytics integrates with Azure IoT Hub and Azure IoT Suite to enable powerful real-time analytics of IoT sensor data. It seems not a week goes by without another announcement of yet another useful microservice from Azure, AWS, Google, and IBM.

Figure 9

Polyglot Cloud Deployment

C3 AI Suite's model-driven architecture provides polyglot capability, enabling application portability from one cloud vendor to another and the ability to run AI and IoT applications on multiple clouds simultaneously.



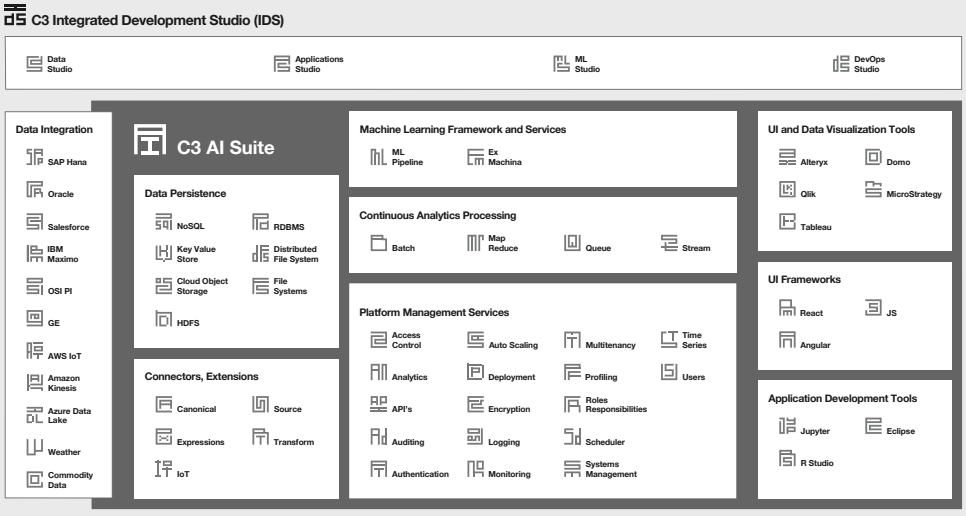
Conclusion: A Tested, Proven AI Platform

The model-driven approach to developing enterprise AI and IoT applications using the C3 AI Suite has been tested and proven in dozens of large-scale, real-world deployments at some of the world's largest organizations, including Royal Dutch Shell, 3M, Enel, and the US Air Force.

Figure 10

C3 AI Suite: A Comprehensive AI and IoT Development Platform

The C3 AI Suite provides a complete and comprehensive set of tools and services necessary for developing and deploying AI and IoT applications at enterprise scale.



The C3 AI Suite provides a uniquely powerful platform enabling these and other leading organizations to develop and operate enterprise AI and IoT applications at scale, with a fraction of the effort and resources required by other approaches. Applications built with the C3 AI Suite are flexible, easily upgraded, and can be ported across different cloud platforms with little or no modification, providing a solution that future-proofs customers' investment in enterprise AI and IoT application development.

About C3.ai

C3.ai is a leading AI software provider for accelerating digital transformation. C3.ai delivers the C3 AI Suite for developing, deploying, and operating large-scale AI, predictive analytics, and IoT applications in addition to an increasingly broad portfolio of turn-key AI applications. The core of the C3.ai offering is a revolutionary, model-driven AI architecture that dramatically enhances data science and application development.

Proven Results in 8-12 Weeks | **Visit c3.ai/get-started**

C3.ai 1300 Seaport Boulevard, Suite 500, Redwood City, CA 94063