2019
Addressing state agency needs for modernized child support calculators
Considerations for building cost-effective, scalable, and user-friendly child support guideline calculators leveraging IBM technologies
Determining the right tools to support strategic priorities at the right cost can be daunting for child support agencies seeking to modernize their guideline calculators. **We can help.**
A child support guideline calculator (referred to as “calculator” throughout) is a critical service component of a state government’s child support enforcement program, but agencies face challenges efficiently building and deploying calculators that meet government compliance requirements and provide a modern and intuitive end user experience. These calculators are usually part of larger child support case management applications—most of which are monolithic in nature.

A monolithic child support case management application is typically built as part of a singular autonomous system. Making changes to the application—like maintenance and enhancements—can be time-intensive and expensive, as updates typically affect the entire system. Any modifications or changes made to small segments of code can result in the need to build and deploy a completely new version of the software. In addition, updates to accommodate an increase in processing volume or additional user profiles can be problematic, as these entail scaling the entire application. This becomes increasingly complex for the agencies managing child support guideline calculators, as policies and calculation formulas are constantly in flux due to regulations and tax laws.

The typical end users of these calculators are individuals who make and/or receive child support payments and the case workers who generate estimated child support benefits for court orders. The user experience of these calculators is highly dependent upon the accuracy of parameters provided by end users. Existing calculators require a multitude of accurate data points from the end user that must comply with several federal and state policies. This requirement often results in a patchwork of complex and ambiguous software modifications that produce inaccurate child support calculation results—and ultimately render a poor customer experience for the end users.

To keep up with continual changes in regulation, it is recommended that the state agencies leverage a microservices-based platform, which is well-suited to support this type of application. A microservices-based architecture allows developers to build the calculator’s components as a suite of small services, each running its own process that can be deployed independently. Microservices allow developers to build scalable, flexible applications, which in turn help to reduce the complexity of the calculator making it easier to maintain and enhance.

IBM provides enabling technologies and tools to enhance child support calculators. Pairing a microservices-based architecture with a decision engine such as IBM Operational Decision Maker (IBM ODM)—and operationalizing artificial intelligence (AI) with IBM Watson® and IBM POWER9™ AI servers—can modernize the technology stack that supports the calculator and improve end user experience.

This paper investigates an innovative approach to modernize the legacy architecture of a child support guideline calculator. It explores the use of containerized microservices architecture with isolating business rules of the calculator using IBM ODM on the IBM WebSphere® Liberty platform. It also explores using intelligent optical character recognition (IOCR) capabilities with IBM® Datacap, and chatbot functionality using IBM Watson on AI-enabled IBM POWER9 servers for process automation and an enhanced end user experience.
Business drivers

Several market influences are prompting state child support agencies to modernize their mission-critical legacy applications, including child support calculators. State agencies can leverage IBM technologies to help alleviate various challenges associated with modernizing monolithic applications and improve the maintainability, scalability, and agility of these calculators. Additionally, microservices-based architectures that incorporate IBM technology can provide end users with improved front-end calculation experiences, and help states adapt to rapidly changing regulatory requirements.

Improve user interface/user experience (UI/UX)

A typical calculator can be archaic in nature since states usually require the end user to input a variety of parameters such as income, existing child support payments and other household information. In many instances, the end users either do not have the knowledge necessary to provide these inputs, or do not know how and/or where to obtain the information needed for a child support guideline calculation. As a result, many states provide an estimation or simplified versions of these calculators which produce incorrect child support computations. States must also account for shifts in calculator user demographics. End users of such calculators—especially case workers—are usually younger, tech savvy individuals who expect a modern human-centered experience.

It is vital that states create better UI/UX experiences in order to effectively operate child support enforcement programs. Several IBM technologies can be leveraged to support this mission. For example, end users are required to enter information from a prior year’s digitized tax return form. This step can be avoided with automated pre-population by leveraging an IOCR technology like IBM Datacap. Mobile functionality is also important for increased UI/UX. Chatbots and auto-text leveraging artificial intelligence (AI), such as IBM Watson, can help states create an intuitive user experience and greatly reduce manual data entry and human error.

Increase maintainability and scalability

These calculators are typically built as part of monolithic applications that are singular and autonomous units. Maintaining existing platforms and accommodating regulation changes is expensive due to the amount of time needed to make changes and the highly-technical specialized resources required. Making these modifications can result in regression testing and the need to deploy the entire application, making it difficult to achieve the automation and scale required for efficient DevOps. Scaling specific components on a monolithic application also means scaling the entire application. This fosters an environment which is change-resistant—a difficult prospect in the public sector where budgets are limited and resources with specialized skillsets can be hard to find.

The right architectural pattern, specifically one that is microservices-based, can help to address these challenges by improving the maintainability and scalability of the applications. IBM technologies such as the IBM WebSphere Liberty platform and IBM ODM have the inherent features to improve the maintainability and scalability of the applications, and are better suited for enabling a microservices architecture pattern.
Increased regulatory compliance
Today, government organizations require greater agility to meet regulatory requirements than ever before. In particular, child support programs are federally mandated and must meet strict guidelines in order to avoid penalties and best meet the needs of citizens. Legacy solutions based on large, monolithic code structures and tightly coupled programs typically do not adapt to the fluidity of industry regulations.

Typical child support applications require years of custom coding and have limited integration capabilities, making it difficult to increase straight-through processing that addresses changes in regulation. Modernizing from a monolithic architecture enables public agencies to have a responsive and well-integrated system to respond to rapidly changing requirements.

Improving the agility of applications allows state agencies to better meet increased regulatory guidelines and keep up with the pace of policy change. A microservices architecture enables public agencies to have a responsive and well-integrated system to respond to rapidly changing requirements.

Increase agility and reusability
Developers must take several steps to change monolithic applications before any changes to a calculator can be deployed. A thorough analysis must be conducted to examine the impact during build, system testing, and deployment. This can be a time-consuming process, especially if the proposed changes are minor. In addition, the reusability of components in a monolithic application is limited because the application is built on a singular autonomous system.

A microservices architecture improves the agility of the applications by enabling developers to modify and deploy only the services that require modification without affecting other services. Individual services can also be easily modified and scaled appropriately depending on business needs. The IBM WebSphere Liberty platform on IBM LinuxONE offers a cost-effective application server and hardware platform to augment the microservices architecture for the calculator. Additionally, IBM® API Connect with IBM DataPower provides a secure API platform that increases the agility so that the systems can communicate with each other.

Increase self-service for business logic changes
Child support guideline calculators have complex business logic that include multiple formulas and various decision points based on parameters such as tax year, state, number of dependents, and more. If an end user needs to make a small change to the parameter calculation, they may require assistance from a software developer. This developer, in turn, might need to review thousands of lines of code to identify where the change is required. This process is becoming increasingly complex as there are fewer resources available who are able to review and update legacy code.

As such, there is an increasing need for a business analyst to test “what if” scenarios in order to make these business logic changes without requiring software developers to analyze thousands of lines of code. This type of decision engine calls for a “shift left” logical pattern, which works well with a microservices architectural pattern using a business rule engine like IBM ODM.
Architecture overview

The latency that results from changing monolithic applications is significant. This can be especially challenging in today's agile environment in which delivery windows are small and continuous in nature. There is a way to transition a child support calculator application from a monolithic to a microservices architecture pattern.

### Characteristics of current and future state calculator architectures:

<table>
<thead>
<tr>
<th>Reduced complexity for improved user experience</th>
<th>Monolithic architecture (current state)</th>
<th>Service oriented architecture (current state)</th>
<th>Microservices Architecture (future state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolithic applications can require large amounts of effort for regular maintenance and upkeep</td>
<td>Although SOA applications are simpler than monoliths, they are not as light as microservices</td>
<td>Logic contained in small microservices is easier to deploy, run and maintain</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Improved agility and reusability</th>
<th>Limited reuse is realized across monolithic applications</th>
<th>SOA improves reuse potential as compared to monolith</th>
<th>Microservices can be easily reused for multiple use cases</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Increased scalability</th>
<th>Scaling monolithic applications can often be a challenge</th>
<th>SOA can be scaled up easily as compared to monoliths, but is still more complex than microservices</th>
<th>Scalability can be achieved easily, especially if the services are containerized and orchestrated using Kubernetes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Improved maintainability via streamlined DevOps</th>
<th>Achieving operational agility in the repeated deployment of monolithic application artifacts is difficult</th>
<th>DevOps and continuous delivery (CD) is becoming popular, but not yet mainstream</th>
<th>Modern DevOps practices of continuous integration (CI) and CD can be easily applied to this pattern</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Technology stack</th>
<th>Monolithic applications are implemented using a single development stack (i.e., JEE or .NET), which can limit the availability of “the right tool for the job”</th>
<th>SOA can be implemented in any suitable technology as it exchanges information in predefined format, such as XML</th>
<th>Individual services can be implemented in any suitable technology, as information is exchanged in a pre-defined format such as JSON</th>
</tr>
</thead>
</table>

**CURRENT STATE**

**Monolithic architecture**

- UI Layer
- Business Layer
- Utility
- Arch
- Financial
- Database

**SOA architecture**

- UI Layer
- API Layer
- GC
- Database

**FUTURE STATE**

**Microservice Architecture**

- UI Layer
- API Layer
- GC1, GC2, GC3, GC4
- Database
CURRENT ARCHITECTURAL PATTERNS

Monolithic architecture
A typical monolith application is multi-tiered in nature. The front-end presentation layer usually consists of the user interface, the middle tier contains the business logic, and the back end hosts a database storing all the required data for the application. The child support guideline calculation logic is buried in the middle tier and the reference data needed for this calculation is part of a huge, back end, single database.

To make any alterations to the system, a development team must build and deploy an entire server-side application. Thus, the development, testing, and deployment effort becomes time consuming, costly, and prone to errors.

SOA architecture
Some agencies have upgraded their calculators to service oriented architectures (SOAs). With SOA, the business functionality is grouped together and exposed as-a-service. SOAs provide significant improvements from monolith architectures.

The service may have multiple methods and interfaces to achieve the desired aspects of the functionality. SOAs typically rely on single databases with complex schema to cater to the data architecture needed for the appropriate functionality. SOAs remove several of the hassles of monolith architecture, but do have some limitations.

FUTURE STATE ARCHITECTURAL PATTERN

Microservices architecture
In a microservices-based architecture, a child support calculation logic is divided into several microservices that coexist as independent entities. Microservice capabilities are expressed formally with business-oriented APIs. The implementation of the service is completely abstracted as the interface is defined purely in business terms. The dependencies between services and the end consumer of services are minimized by applying the principle of loose coupling. Standardizing contracts expressed through business-oriented APIs means that consumers are not impacted by changes in the implementation of the service. This allows service owners to change the implementation and modify the systems of record or service compositions which may lie behind the interface and replace them without any downstream impact. The IBM API Connect platform is an enabler of this functionality. While a microservices pattern has its own set of challenges, the potential benefits in this case outweigh the drawbacks.
Architecture layers

A multi-layered software architecture consists of various layers that each correspond to a different service or integration. Making changes to an individual layer is simpler than having to update the entire architecture. In addition, making updates to legacy systems is easier when the architecture is broken up in this fashion.
PRESENTATION LAYER

The UI/UX layer of a calculator is the top layer in a multi-tier microservice architecture. Any web stack technologies such as Angular, .NET frameworks, or React library can be integrated to develop responsive web applications and provide an intuitive user experience. This UI should be developed as a separate service module and deployed in an IBM WebSphere Liberty server container. An end user can then access this through any type of a web browser. This separation of layers gives development teams the ability to develop and enhance the specific UI/UX layer with minimal impact on the other layers and greater speed than developing a singular code base.

The presentation layer communicates with the business logic layer under it, passing the information from the user and controlling it, then giving back any response it produces. This interaction is illustrated in the graphic to the right.

An IBM® Watson Assistant chatbot can be coalesced into the user interface to provide a better user experience. The AI chatbot can be trained and configured with intents, entities, and design flow to support guideline calculations. While using the calculator, users can be prompted to use this chatbot when the application is opened in the browser. It can also be trained to guide users to provide accurate information for the calculation.
INTEGRATION SERVICES LAYER

Managed in the integration layer are API creation, management, enforcing user policies, authentication, authorization, and usage and performance analysis of APIs.

Enterprise API users can be internal, partner agencies, or the public. In all cases, providing access to enterprise APIs presents challenges such as security, auditing, measurement, billing, and lifecycle. API management aims to provide a unified approach for addressing challenges like:

- Rapidly creating new APIs from existing business assets or cloud services through configuration and a no-coding approach
- Taking a “one stop shop” approach to facilitate service discovery
- Securing, controlling, and optimizing access to APIs
- Managing APIs with business-level controls by setting varying levels of consumption entitlements and managing the application developer
- Accessing detailed analytics on the usage for each API
- Reducing resolution time with supplied operational monitoring and debugging capabilities

The calculator provides two tiers of APIs: a free tier API (which would be accessible to the public per state mandate), and a paid service for more accurate calculation of child support using AI capabilities. The API monetization aspect is handled by the API gateway and can be implemented by IBM API Connect platform.
SERVICES LAYER

This central layer of the model deals with the logic of the program. It receives data from the upper level and transforms it, using the inner application logics. It also retrieves data from the deepest data layer and uses it. By integrating these two processes, it performs modifications in both levels.

The services layer contains the determinant part of the application logic, including:

- Performing all required calculations and validations
- Managing workflow
- Keeping track of application execution
- Distinguishing among application instances through session management
- Identifying users
- Providing access to application services in a consistent way
- Managing all data access for the presentation layer

As illustrated in the above graphic, an incoming request from the UI chatbot invokes a REST endpoint developed in the backend microservice to pass the JSON payload. Backend microservices are built using spring boot cloud framework and are integrated with IBM Watson SDK libraries. IBM Watson libraries enable backend service to handle asynchronous REST calls from the chatbot. This service uses a containerized IBM WebSphere Liberty server.

Every request received from users is captured into a parameterized JSON object, which is then sent to IBM Watson. IBM Watson Assistant calls the backend service using Gateway API to render a response on the user interface. Each of these server-side microservices are bound in service discovery. This service discovery redirects request to designated REST endpoint for processing.
RUNTIME SERVICES LAYER

Containerization
A microservices architecture uses containerization to make the solution scalable, portable, and highly available. All basic building blocks of each service (e.g., Spring Boot, IBM WebSphere Liberty Server, and PostgreSQL) are containerized using Docker framework.

Since the microservice is relying solely on reference data and does not need any transactional support, the reference data is stored in a JSON configuration file and embedded in the base docker image. A new version of the image is created and pushed to the container registry if the reference data is changed.

Kubernetes is used as a container orchestration framework for container management, deployment, and scalability. It also performs other tasks such as health checks, self-healing, traffic routing, and load balancing. A containerization framework helps with the following aspects:

- **Process isolation**: All microservices are running in their own process and catastrophic failure in any service won’t bring down other services
- **Scalability**: Each service’s resources can be scaled up or down as per the load requirements using out-of-the-box Kubernetes functionality
- **Portability**: Even though this application stack is focused on the IBM® LinuxONE™ platform, the containerized services can be easily ported to any platform which offers docker and Kubernetes framework components
BUSINESS SERVICES LAYER

The business service layer plays a vital role when a user engages with advanced services on a calculator, such as submitting tax return documents from previous years. The service layer microservice sends an encrypted file to the business service layer to get a data transfer object for processing. The core orchestration of a REST end point is also triggered from this layer. Based on the request, this layer determines the designated end point responsible for processing the request and provides a response. Information comes from the upper level for computation and transformation and is sent back to the upper layer.

IBM Datacap can support the business need to transform file data into an object for processing, such as tax return files.

AI with IBM Datacap
This backend service should also be integrated with IBM Datacap, which is an IOCR service for text extraction from the file. In this use case, end users need to upload four years of previous tax return files utilizing the UI or chatbot to accurately compute child support payments. IBM Datacap transforms the file content into a marshaled object, which is sent to IBM Watson for prediction in the form request payload. IBM Watson triggers an event to the backend service to fetch tax bracket information, along with other input parameters, so that the state can perform a calculation based on past and current data. An analytics platform such as IBM Cognos Analytics can be used to visualize the predicted result to the calculator user. Whenever a user submits a request for processing, a new data confirmation page is displayed with pre-populated records from the file. Once a user confirms the data, the request is sent to IBM Watson for a child support amount prediction.

AI with IBM ODM
Child support calculation results are derived from the combination of multiple child support policies. These policies become challenging to maintain when they are integrated in SOA or monolithic applications. Implementing and bundling these policies into IBM ODM can provide easy integration with microservices and reduce IT overhead when changes to these policies are needed.

An IBM ODM microservice can be developed using IBM Rule Designer SDK. This microservice is published into the IBM Decision Center and deployed into an on-premise IBM WebSphere Liberty server (Decision Server). The IBM Decision Center works as a separate instance in a container, which provides flexibility to users with administration or configuration manager rights access to create and update deployed decision configurations on-demand. All rules related to the calculator reside in this service, but the actual business logic is implemented in the core microservices.

The request payload is sent to ODM in JSON format from the UI client using ODM's REST endpoint. Based on the payload, ODM determines the rule policy to orchestrate the microservice REST endpoint where the request payload is processed, and a response is sent back to the user.

To illustrate when a user submits information to calculate a guideline support amount, these configured rules identify and collect basic information and trigger an event for completion (which includes their demographics details, income, state of residency, etc.). Once that information is available, ODM determines the corresponding core microservices REST endpoint to process the request payload. However, an HTTP request is not redirected to ODM when users have submitted a request from the chatbot or submitted tax return files.
Modernizing with help from Deloitte and IBM

Determining the right tools to support strategic priorities at the right cost can be daunting for child support agencies who are seeking to modernize their guideline calculators. These applications are typically monolithic in nature, and modernizing them often results in many challenges such as inability to implement time-sensitive changes, inability to maintain in a cost-effective manner, and an archaic end user experience. This reference architecture details the IBM technologies that agencies can leverage, as well as an accelerated path that can be applied to child support guideline calculators and similar tools.

IBM provides enabling technologies and tools that are well-suited to modernize such applications. Pairing a microservice-based architecture in conjunction with these enabling IBM technologies such as a decision engine such as IBM Operational Decision Maker (IBM ODM)—and operationalizing Artificial Intelligence (AI) with IBM Watson® and IBM POWER9™ AI servers—can help alleviate many of the challenges outlined in this paper.

The proposed architecture and supporting IBM tools and technologies can be easily implemented as a hosted service or service instance of the calculator. IBM technologies are enablers for a modernized microservices-based architecture, can help reduce technical debt, manage unsupported cross-platform support, and help reduce complex software ecosystems that can be difficult to troubleshoot.

For this specific use case, leveraging products from a single vendor can help reduce friction, risk, and cost during critical periods of transition for business and IT. IBM offers many hardware and software products that support modernization objectives, ranging from platform options to security products and middleware tools—which can all be used to alleviate pain points during initial modernization and transitional periods throughout the process.
IN ACTION: Building a cost-effective, scalable calculator for a state child support agency

CHALLENGE
Due to a federally mandated child support program, each state needs to provide its citizens with a calculator to streamline and understand guidelines around child support payments. Some states require more robust calculators than others in order to generate more accurate payment calculations. Typically, child support calculators are built into monolithic applications on legacy platforms or on traditional SOA patterns.

In this representative example, a large state government is seeking to modernize its child support guideline calculator using the architectural patterns and the IBM technologies mentioned in this paper. This state is required to provide a robust calculator that requires multiple intra-agency approvals before release. The state faces challenges implementing changes quickly and validating them through the certification process.

The monolithic nature of the calculator and the lengthy certification process resulted in implementation delays and slowed the proposed release cycle of the calculator. In addition to updating the legacy architecture, the state needs an improved end user experience to provide a more intuitive calculator that will yield more accurate child support guidelines. Finding resources with the specialized skillsets needed to support child support guidelines is difficult, and the state lacks dedicated staff who could assist end users through live chat or via phone.

Overall, the state needs a more robust and agile release cycle and self-service portals to enable the deployment of modernized child support guideline calculators. Other crucial requirements include the ability to adapt quickly to changing federal and state requirements, to improve end user experience though UI/UX and the ability to be scalable and maintainable. The state hopes to solve its technical and resource-based challenges by modernizing the architecture of its child support calculator to a microservices platform and implementing overall process changes.

Approach
The state requires an incremental approach to upgrading the calculator architecture due to budgetary and technical resources constraints. Deloitte works with the state to develop a roadmap for updating the calculator’s monolithic architecture.

Additionally, the state prefers to use technologies with which their technical resources are familiar so internal teams can continue implementing and maintaining the architecture of the child support calculator. The state had already adopted an agile methodology which was conducive to updating the monolithic architecture to a microservices-based architecture.

Process
The first phase in the roadmap involves modernizing the business layer of the calculator, which requires multiple incremental steps. More than 25 compute formulas need to be refactored into microservices, followed by 25 decision points that needed to be moved to a rules engine. More than 50 database reference tables can be split according to the microservices that they supported. End user experience can be modernized by using human-centered design principles for the calculator portal, making it more responsive and providing a chatbot capability. This requires a refactoring of several screens of the calculator. Deloitte would guide the state throughout the process due to our experience in modernizing legacy applications and implementing IBM technologies.

Impact
Based on the proposed technologies, tools, and reference architecture, it is expected that the state’s current application would be upgraded to a modern, user-friendly child support calculator. The upgraded calculator can provide the state with the necessary agility to release updates on-demand in order to meet necessary child support certification cycles.

The microservices-based architecture and business rules engine can help reduce the complexity of the calculator, allowing the teams to quickly develop, system test, and regression test. The updated architecture enables the agency’s staff to make small changes themselves and enables the development team to focus on more complex changes.

The end user interface of the calculator can be improved significantly. For example, the IBM Watson-based chatbot allows the service desk to focus on more complex system issues and calls because users can discover answers to many questions by using the natural language features of the chatbot. This feature can easily scale to assist a growing number of citizens using the customer portal for their service.
Leading technologies for a child support guideline calculator

**IBM LinuxONE**

IBM LinuxONE is an enterprise-grade Linux server with a unique architecture designed to bring IBM’s experience in building secure, resilient, scalable and high-performance reliable systems with the openness of the Linux operating system. LinuxONE is a Linux-only platform intended to support customers’ interest in leveraging the open source ecosystem combined with highly secure and highly scalable servers. IBM has marshaled key open source and industry software for LinuxONE systems including Python, Go, Swift, Java, and other languages; MongoDB, PostgreSQL, Apache Spark, Node.js, Hadoop, and other tools including Kubernetes, Docker, Chef, and Puppet etc.

IBM LinuxONE systems and solutions provide users with a flexible, powerful infrastructure that helps ensure that businesses and government agencies have the performance, reliability, security, and processing power they need to address increasingly sophisticated and demanding application requirements.

Due to its simple architecture and efficiency, LinuxONE is an economical option compared to competitive products. Additionally, the security and services embedded in LinuxONE provide great value to child support agencies who need to modernize their calculators.

**IBM WebSphere Liberty**

IBM WebSphere Liberty is a high-performing application server that can support the lightweight multi-tier microservice application (based on SpringBoot and Spring Cloud framework) of a calculator. Lightweight application servers such as IBM WebSphere Liberty help to integrate CI/CD and support testing and deployment of web applications in multiple environment deployments.

The Liberty profile is a simplified, lightweight development that contains the subset of Liberty that corresponds to the Java™ EE Web Profile specification. The application runtime environment has several benefits, including:

- **Simple to configure** – Configuration is read from an XML file with text-editor-friendly syntax.
- **Dynamic and flexible** – The runtime loads only what the application needs and recomposes the runtime in response to configuration changes.
- **Fast** – The server starts in fewer than five seconds with a basic web application.
- **Extensible** – The Liberty profile provides support for user and product extensions, which can use System Programming Interfaces (SPIs) to extend the run time.

Since the architecture for a child support guideline calculator is based on a microservices pattern, IBM WebSphere Liberty is a strong platform to monitor application health, trace logs, and handle fault tolerance such as timeout, retry, and more.

**IBM® API Connect with IBM DataPower® Gateway**

IBM API Connect is an industry-leading API gateway. API Connect helps to rapidly publish, govern, socialize, analyze, monitor, and monetize APIs with built-in capabilities at every stage of the lifecycle, from planning, designing, and developing to testing, deploying, and retiring.

IBM DataPower works hand-in-hand with API Connect for added security and scalability. It delivers a robust XML firewall for the enterprise, using sophisticated checks. It also provides deep content introspection at wire speeds and service request and response monitoring support. The DataPower appliance enables the classification of data requests based on service and application-level information.
IBM Operational Decision Manager (ODM)
Considering the complexity of calculator applications, state agencies should consider integrating an enterprise business rules management system to automate critical business tasks. A decision rules system increases the productivity, consistency, and quality of important business processes.

IBM ODM provides several key advantages:

- **Productivity** – Provides flexibility for on-demand updates and deployment, making it easy for business experts to define and understand rules by representing them in natural language, rather than in programming syntax.

- **Collaboration** – Offers the ability to share expertise to define rules more quickly and with higher quality. IBM ODM’s Business Console allows business experts to use social networking mechanisms to quickly identify work of interest and share ideas with colleagues.

- **Governance** – Allows business experts to manage changes for greater security and accountability. IBM ODM Console provides users with the option to create releases, versions, and activities with specific goals and deadlines.

- **Insight** – Offers reporting capabilities to identify gaps and overlaps in rules, as well as to provide greater visibility to business experts and other stakeholders.

In a calculator application, IBM ODM helps to reduce information gathering and overhead processing. It makes applications more intelligent and robust so that they can process complex computations. Integrating IBM ODM Decision Center in the guideline support application provides business analysts with the ability to manage complex rules.

---

IBM Datacap
IBM Datacap is a critical component for leveraging the AI capabilities of the platform and provides these key elements:

- **Increased productivity** – By significantly reducing manual data entry and paper-based storage and retrieval of documents, knowledge workers can quickly process and access documents for customer service, case management, business transactions, and compliance.

- **Streamlined process** – IBM Datacap can help eliminate a cumbersome paper process by enabling clients to automate previously labor-intensive tasks.

- **Reduced cost** – IBM Datacap is designed to reduce the required data entry personnel by 50 percent, which keeps costs down and makes data entry more efficient and cost-effective. Distributed scanning and verification enable clients to reduce or eliminate document shipping costs and distribute labor to areas with more affordable labor rates.

- **Increased accuracy** – By eliminating the errors that human data entry operators make, Datacap Taskmaster Capture can save time tracking down and fixing faulty data or misplaced document images.

- **Flexible rules processing** – The procedural rules engine that powers IBM DataCap uses .NET and VBScript standards which enable clients to create advanced capture applications with custom panels and features that address their specific business objectives.

---

IBM Watson Assistant
IBM Watson Assistant is an industry-leading conversational AI technology powering chatbots. It provides these elements, which are all crucial to the architecture of a child support guideline calculator:

- **Skill routing** – Determines whether it should provide a direct answer or reference search results from a document or database.

- **Hosting flexibility** – Can be deployed in an isolated cloud environment or on-premises.

- **Powered by AI** – Industry-leading AI powers the underlying natural language models and provides training recommendations as you build.

- **Ability to work with unstructured data** – Leverages cognitive computing for analyzing large masses of information and unstructured data.
References


## Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>CI/CD</td>
<td>Continuous Integration/Continuous Deployment</td>
</tr>
<tr>
<td>CPA</td>
<td>Certified Public Accountant</td>
</tr>
<tr>
<td>DevOps</td>
<td>Development and Operations</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IBM ODM</td>
<td>IBM Operational Decision Manager</td>
</tr>
<tr>
<td>IOCR</td>
<td>Intelligent Optical Character Recognition</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>Java EE</td>
<td>Java Enterprise Edition</td>
</tr>
<tr>
<td>OCR</td>
<td>Optical Character Recognition</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SPI</td>
<td>System Programming Interfaces</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>UI/UX</td>
<td>User Interface/User Experience</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Authors

Hemang Dholakia
System Architect
Managing Director, Deloitte Consulting LLP
hdholakia@deloitte.com

Venu Bommenani
Technology Project Manager
Senior Manager, Deloitte Consulting LLP
vbommenani@deloitte.com

Rajiv Patel
Application/Solution Architect
Project Delivery Manager Senior, Deloitte Consulting LLP
rajpatel@deloitte.com

Rajesh Parab
System Architect
Specialist Senior, Deloitte Consulting LLP
raparab@deloitte.com

Santosh Kumar Dubey
Senior Application Developer
Senior Project Delivery Analyst, Deloitte Consulting LLP
santdubey@deloitte.com

Rinku Sinha
Consultant, Deloitte Consulting LLP
rinsinha@deloitte.com

This communication contains general information only, and none of Deloitte Touche Tohmatsu Limited, its member firms or their related entities (collectively, the “Deloitte Network”), is, by means of this communication, rendering professional advice or services. Before making any decisions or taking any action that may affect your finances, or your business, you should consult a qualified professional adviser. No entity in the Deloitte Network shall be responsible for any loss whatsoever sustained by any person who relies on this communication.

As used in this document, “Deloitte” means Deloitte Consulting LLP, a subsidiary of Deloitte LLP. Please see www.deloitte.com/us/about for a detailed description of our legal structure. Certain services may not be available to attest clients under the rules and regulations of public accounting.

Copyright © 2019 Deloitte Development LLC. All rights reserved