



NCCSD Systems Workgroup Vendor Forums – Q&A related to “Low Code/COTS”

Vendor Name: **RedMane Technology LLC**

RedMane would like to thank you for this invitation to share our perspective on this important and timely topic. In our two decades of experience in building and supporting Child Support systems across the country, we have had the first-hand opportunity to witness, participate in, and indeed contribute to the emergence of Low Code/COTS solutions. Through the research and development performed in the development of RedMane’s own case management and Child Support financials COTS offerings, we have seen and learned a lot, and we are pleased to share our observations and lessons learned with you.

Questions:

- 1. Since there is not yet a consistent term or definition for this approach, please give your company's description, including your terminology and definitions. How is this approach different from a "custom" build of a Child Support system? If you choose to do a quick demo or screen shots that would be welcome.**

A FUNDAMENTAL DEFINITION

As we work to describe the term “Low Code/COTS”, let’s start with a foundational definition of the term “COTS” itself. COTS is an acronym that stands for “Commercial Off-The-Shelf” and is generally applied to software products that are ready-made and available for sale to the general public. The Microsoft Office suite is an example of COTS software familiar to most of us, with its specialized components such as PowerPoint, Word, and Excel each providing a rich set of features and functionality right out of the box.

A Low Code/COTS solution provides a configurable feature set of tools to the users, allowing them to create and modify core COTS functionality without traditional computer programming. This toolset is typically represented by a series of graphical administrator user interfaces.

So, now that we have established a starting point for exploring the meaning of Low Code/COTS, let’s go back in time to re-visit a brief history of large software systems development to see how we got to where we are today.

THE JOURNEY TO LOW CODE/COTS – HOW DID WE GET HERE?

There was a time (roughly the 1970’s through the 1990’s) when most large Government software systems - including Child Support systems – were custom developed, starting from scratch each time, in the performance of contracts and projects spanning many years and costing many tens of millions or even hundreds of millions of dollars. Most of the legacy Child Support systems in use today were built this way. It was not unusual for these projects to suffer extensive schedule delays, experience massive cost overruns, and in the end produce systems that failed to yield the envisioned capabilities or the positive impacts so badly needed by the agencies. These systems were often huge, typically made up of millions of lines of computer code written by large teams of human programmers. The systems were unwieldy, hard to maintain, and often quite “buggy” – that is, full of defects and imperfections.

THE EVOLUTION OF COTS TO “LOW CODE”

As the methodologies for building systems were evolving from the classic Waterfall approach to a more incremental, “Agile” style (please see [APPENDIX A](#) for a brief discussion of these methodologies), the world of COTS software was evolving as well. Early COTS software, while offering a foundation of powerful, pre-built functionality, still required massive amounts of custom development work to be performed by large groups of software developers and various other technical product specialists. As a result, the projects undertaken to implement these kinds of systems still tended to be very large, very expensive, and typically took a number of years to complete. Examples of these early COTS systems would include things like SAP Enterprise Resource Planning (ERP) and Siebel Customer Relationship Management (CRM) implementations. These projects have typically utilized a monolithic Waterfall approach, with the resultant, highly disruptive “big bang” deployment into the operational environment.

In recent years, a new breed of COTS software has emerged. Known (or branded) as “Low Code”, these products offer the combination of:

- A robust, feature-rich platform providing a foundational set of baseline functionality (e.g. Case Management);
- The ability to tailor and extend the product’s core capabilities through versatile **configuration** of functionality to meet the organization’s specific requirements. **This concept of “Configurability” is perhaps the most key attribute of Low-Code COTS.** What this means is that, instead of requiring the engagement of large groups of software developers to do a lot of custom code development work on top of the platform, the desired functionality can be produced by non-technical team members – via user-friendly administrative configuration features. A few examples of system functionality that can be configured in this manner would include:
 - **Screen design and modification:** Adding, changing, or deleting data entry and display fields; tailoring of look and feel; overall screen navigation;
 - **Process and Workflow support:** Establishing and updating the flow of user interaction and data handling within the system to accommodate the organization’s unique requirements – both initially and on an ongoing basis as business processes evolve over time;
 - **Reporting and Analytics:** Dynamic design and maintenance of static and ad hoc reports, visual dashboards, and related utilization of system data available for enhanced decision support and analysis.
 - **Domain-specific functionality:** Low Code/COTS products that have been purpose-built for domains such as Case Management will provide a tailored set of configurable capabilities to support – in the example of case management – areas such as case establishment, prioritization, assignment, rules for sharing and securing cases, routing and workflow, approvals, and case closure.

LOW CODE EXAMPLES

Below we have provided a couple of examples on the ease of using a Low Code COTS solution. We have included in this section a “before” screen image of a Low Code COTS solution. A second image will demonstrate the administrator configuration tools for that screen and the ease with which the “before” screen can be modified. The last “after” screen shows the applied change in action. All this can be achieved without any programming change or re-deployment – in other words, with “No Code”.

Example 1: Adding dashboards to the homepage

Our first example demonstrates the ability to change your user dashboard. In this scenario, the “before” image shows the available functions for a user on their home page. We want to modify this homepage by adding a few dashboard elements to show Obligations by Type, Cases by Case Status, and Court Orders by State.

The Original (“Before”) System Homepage

Figure 1 shows a screen shot of the homepage before any changes are made.

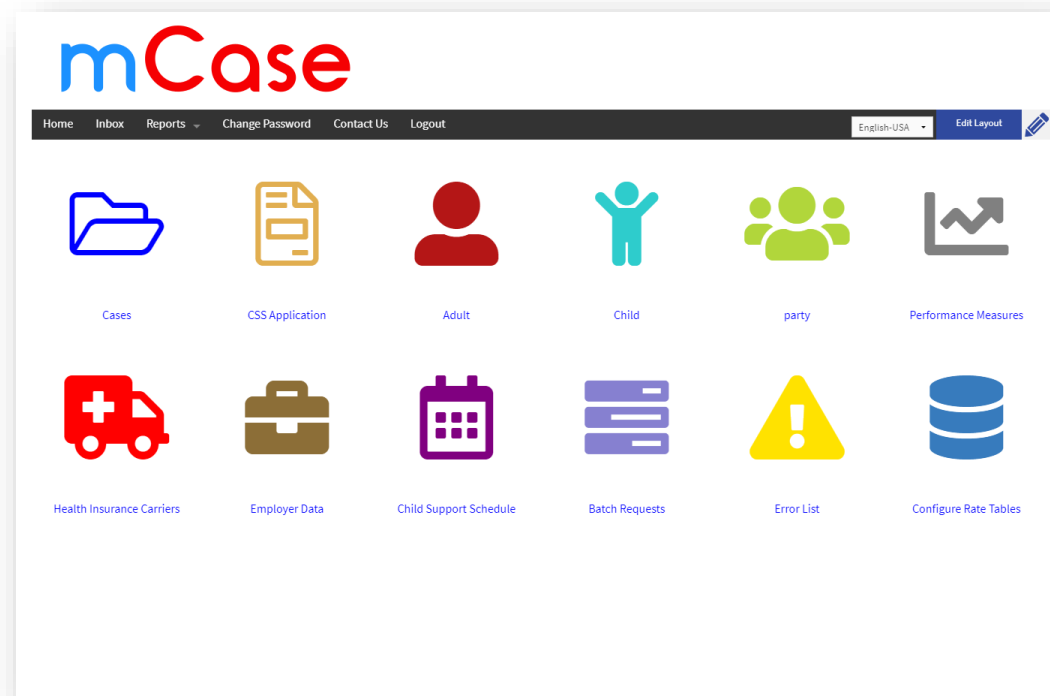


Figure 1: Original (“Before”) System Homepage

Configuration Change Screens

The screen in Figure 2 shows the configuration change being made via the user-friendly configuration interface, to add the dashboard for Obligations by Type:

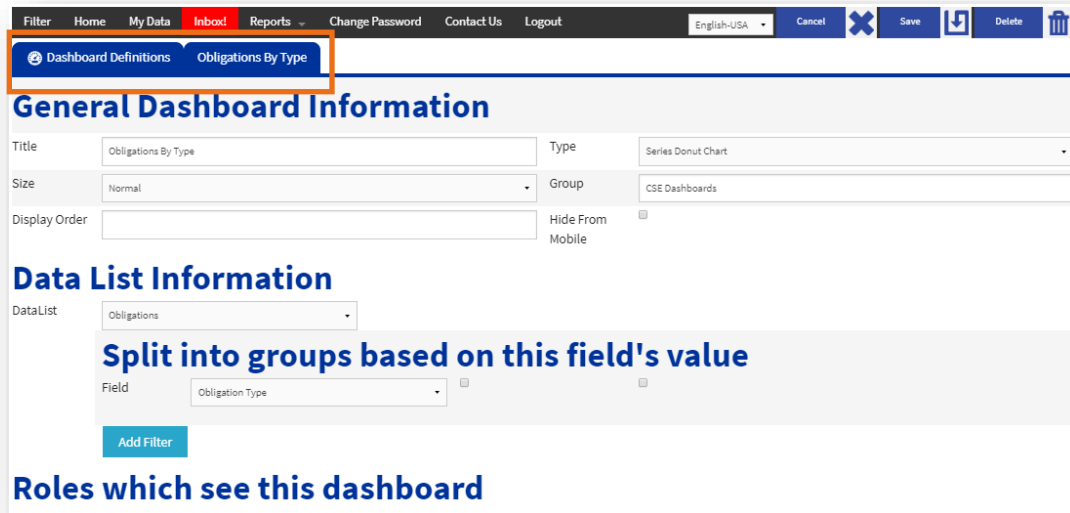


Figure 2: Configuring a new Dashboard element to show “Obligations by Type”

The next screen, in Figure 3, displays the configuration change being made to add the dashboard for Cases by Case Status. A Court Orders by State dashboard could be similarly configured.

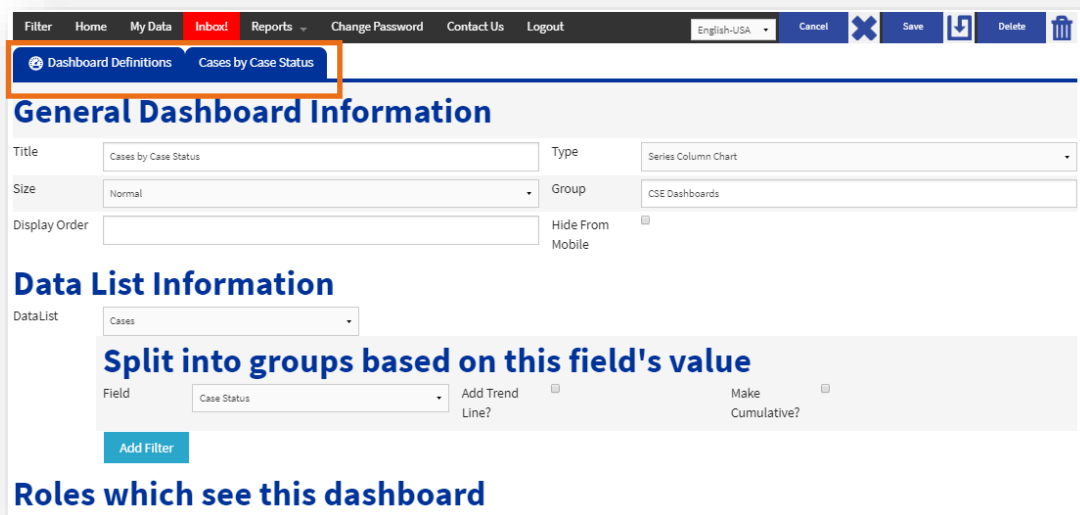


Figure 3: Adding a new Dashboard element to show “Cases by Case Status” information

New (“After”) System Homepage

The new homepage screen seen in the dashboard configurations are applied.

Figure 4 illustrates the resulting homepage after



Figure 4: New (“After”) System Homepage with added Dashboard elements

Example 2: Adding a column to a summary list screen

This next example demonstrates the ability to add (or change) additional information on one of your existing user interface screens. For instance, let’s say we want to add a column and data field showing the “Order Filing Date” to our Court Order information screen.

Prior System Order Summary List

In Figure 5, we have a screen which displays court order information for a case.

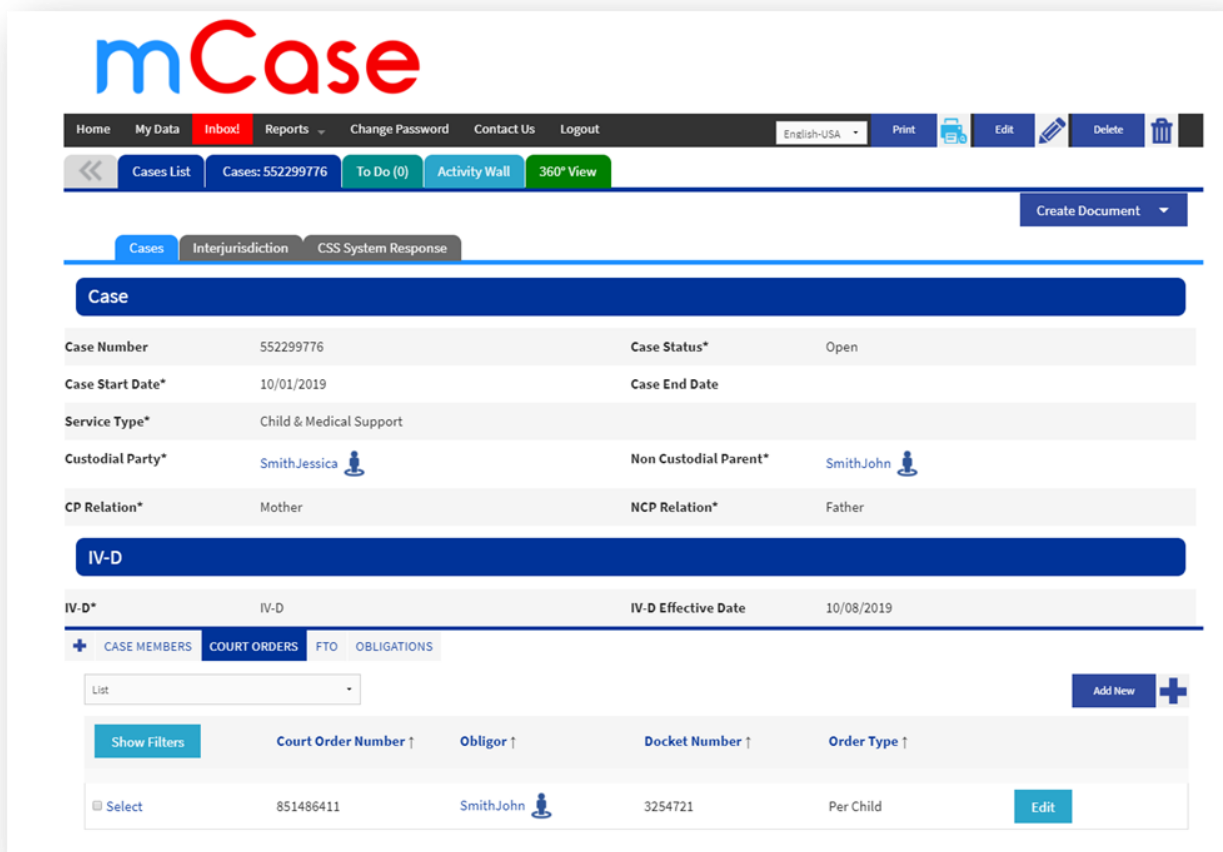
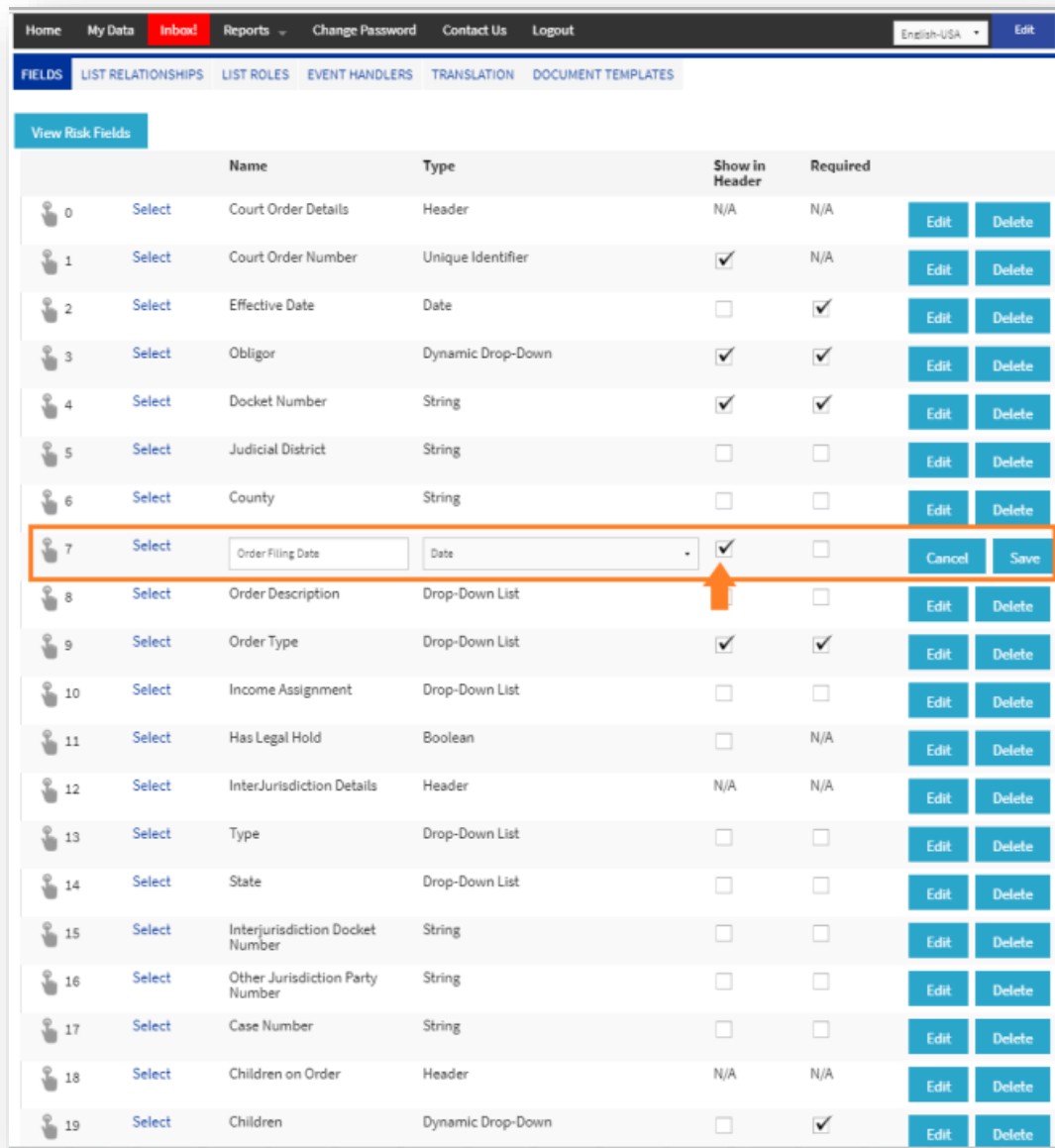


Figure 5: Screen showing Court Order information

Configuration Change

The screen in Figure 6 shows the configuration change being made to add the column and data item “Order Filing Date” to the screen.



	Name	Type	Show in Header	Required		
0	Select Court Order Details	Header	N/A	N/A	Edit	Delete
1	Select Court Order Number	Unique Identifier	<input checked="" type="checkbox"/>	N/A	Edit	Delete
2	Select Effective Date	Date	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Edit	Delete
3	Select Obligor	Dynamic Drop-Down	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Edit	Delete
4	Select Docket Number	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Edit	Delete
5	Select Judicial District	String	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
6	Select County	String	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
7	Select <input type="text" value="Order Filing Date"/>	<input type="text" value="Date"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Cancel	Save
8	Select Order Description	Drop-Down List	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
9	Select Order Type	Drop-Down List	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Edit	Delete
10	Select Income Assignment	Drop-Down List	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
11	Select Has Legal Hold	Boolean	<input type="checkbox"/>	N/A	Edit	Delete
12	Select InterJurisdiction Details	Header	N/A	N/A	Edit	Delete
13	Select Type	Drop-Down List	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
14	Select State	Drop-Down List	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
15	Select Interjurisdiction Docket Number	String	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
16	Select Other Jurisdiction Party Number	String	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
17	Select Case Number	String	<input type="checkbox"/>	<input type="checkbox"/>	Edit	Delete
18	Select Children on Order	Header	N/A	N/A	Edit	Delete
19	Select Children	Dynamic Drop-Down	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Edit	Delete

Figure 6: Configuring the “Order Filing Date” Column and Data Field

New System Order Summary List

In Figure 7, we have the updated screen with the new Order Filing date column and data field added. This was all accomplished in a matter of minutes.

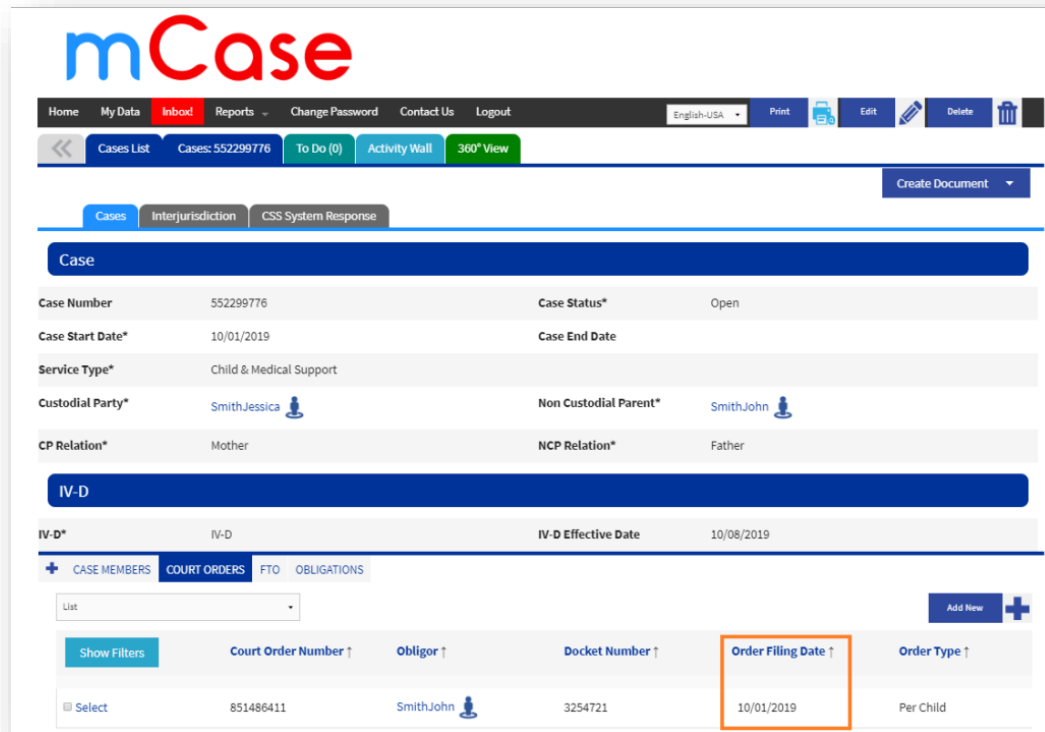


Figure 7: Court Order screen with the new “Order Filing Date” column and data

SUMMARY

To summarize, a Low Code/COTS solution can be thought of as a versatile, highly adaptable software product that provides a pre-built platform and a baseline set of functionality. This core capability is then tailorable to an organization’s specific needs via a rich set of intuitive configuration tools - as opposed to requiring large amount of custom software code development. Hence the term “Low Code”.

2. **With reference to the "core" functionality required by the OCSE Systems Certification Guide (Case Initiation, Locate, Establishment, Case Management, Enforcement, and Financial Management), how does this approach handle each area? In particular, since Child Support requires complicated financial processing, e.g. distribution rules and arrears calculations, please address how these are handled with this approach.**

LOW CODE/COTS FOR CORE CHILD SUPPORT FUNCTIONALITY

A versatile Low Code/COTS solution **purpose-built for Case Management** can serve as an excellent foundation to handle many of the CSE functional areas, such Case Initiation, Locate, Establishment, Case Management, and Enforcement. A purpose-built Case Management COTS solution shares the key building blocks of cases from a structural perspective (cases, people, relationships, effective dating) and they will also include most of the functions that go with case management COTS such as workflow, caseload management, reminders, approvals, calendaring, governance structures and the like.

In its baseline “out-of-the-box” configuration, a given Low Code/COTS solution will typically provide the majority - but not necessarily all - of the exact functionality that a particular jurisdiction (state, territory, county) requires, and this is where the **Low Code configurability** becomes a key part of the solution. It is also important to note that low-code configurability is valuable not just for the initial implementation of the system but is also vital for ease of adapting to changes over time – which has both time and total cost of ownership (TCO) benefits.

LOW CODE/COTS AND CHILD SUPPORT FINANCIALS

Financials are the heart of a Child Support system. Ideally, your Low Code/COTS solution should be able to perform the fundamental financials processing required by Child Support. This processing should be as configurable as possible as this relates to pass through/disregard, interest, DRA fee and threshold, the allocation hierarchy, future holds, etc. The true configurability of these financials functions will enable your state to address future local and federal legislative changes without having to make coding changes. If the Low Code/COTS solution has not addressed Child Support financials, this area will be a heavy lift and most likely the most challenging component in your modernization journey. Consider a financial COTS module as a major subsystem, containing the complexity of your Child Support financial management and processing. Ideally, off the shelf functionality comes pre-configured with the core Child Support financial management functionality provided right out of the box.

In our view, Financial Management in Child Support systems falls into a different category than the other functional areas that you listed. The others are well-aligned and well-served by case management focused Low Code/COTS solutions, but Financial Management is much better served by a true financial platform – such as RedMane’s mFinancials for Child Support - providing foundational elements such as general ledger, accounts management, audit and logging, security, reporting and payment processing interfaces; as well as a Child Support-specific data model including elements such as Party, Case, Obligation, Payment, and



Adjustments.

LOW CODE VS. THE MYTHICAL “NO CODE”

Complete end-to-end Child Support systems, even those based on the most well-suited case management Low Code/COTS solutions, will inevitably require a certain level of actual software code development. For example, some functions of Child Support case management rely heavily on interfaces with external systems and the processing that surrounds them. We see this in the Locate and Enforcement areas especially, but also in Case Initiation and Establishment as well.

A fully functional Child Support system often requires data interfaces with over twenty external systems, and these interfaces typically necessitate a certain amount of custom code development. This absolutely does not invalidate a Low Code/COTS approach, but this is the reason to think of these solutions as “Low Code”, but not “No Code”. All the benefits of pre-built functionality are still there, with the capability to configure the solution to meet local regulations and policy. However, it is important to recognize that there is still some hard work to be done in areas like interfaces and conversion from legacy systems.

CHOOSING THE RIGHT LOW CODE/COTS SOLUTION

The selection of your Low Code/COTS solution or group of COTS products needs to focus both on the Out Of The Box (OOTB) functionality and the ease with which you can configure the solution for the unique needs of a specific state. If more than one COTS product is the best way to provide a complete solution (as is often the case), then additional dimensions such as interoperability and integration of products must be considered as well. You will want to establish a vision of what the final user experience will look like with the integration of these products. The closer the baseline (out of the box) solution comes to satisfying the state and OCSE certification requirements, the smaller the job will be to achieve the final delivered system. A product assessment and gap analysis are typically conducted to determine the magnitude of this change.

SUMMARY

In our view, a Low-Code/COTS solution is indeed a viable approach for Child Support systems. We also believe that the paired, compatible combination of at least two essential Low Code/COTS components – one designed to provide the foundation for case management and another designed to provide the foundation for financials – is likely to offer the best prospects for success.

3. What COTS or other products are used in conjunction with this approach to give a state a fully functional system?

Figure 8 is a diagram representing what RedMane sees as a fundamental representation of a Child Support system’s modular building blocks. These are specific modules for a Child Support Solution that can be leveraged and delivered in a phased, incremental approach. The traditional components are your Case Management COTS, Financial COTS, and Business Intelligence COTS; which make up your core solution. These components must work well together, with the objective of providing a seamless user experience across all functions as well.

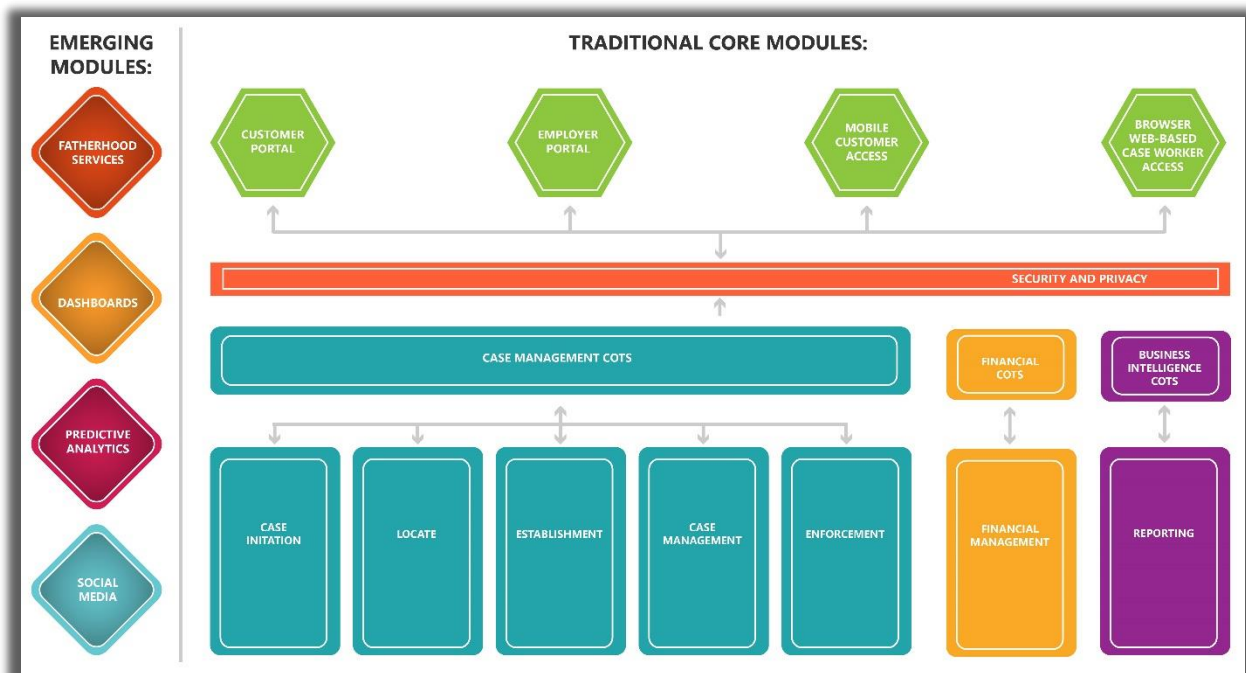


Figure 8: A Modular view of a modern Child Support solution

Additional technology tools and products can be leveraged to extend functionality when delivering additional modules or perhaps to drive an incremental modernization approach. By moving toward a modular COTS approach, states can build on their system incrementally over time in a phased and controlled manner.

It is also important to note that a modular Child Support system can take advantage of technology investments that your state and/or agency has already made and can leverage existing COTS tools - such as business intelligence/reporting solutions, electronic document management systems, and single sign-on tools - to minimize additional procurement costs.



In summary, a sound initial approach for envisioning the total solution and identifying its components would include:

- Establishing a modular functional view of the overall system capability that includes the core elements of:
 - Case Management (including case establishment, locate, paternity establishment, ongoing case management, enforcement, and other Child Support-specific functions)
 - Financials (general ledger, accounts management, and payment processing interfaces, party, case, obligation, payment, and adjustments)
 - Business Intelligence (reporting, analytics, dashboards)
- Working with your Information Technology (IT) organization to determine what tools and technologies may already be available (already purchased/licensed (paid for) and supported by IT) to be part of the new solution vs. the identification of new components (e.g. the Low Code/COTS products) that would have to be acquired;
- Using the framework of the modular system view as a way to envision and lay out a roadmap for the incremental retirement of legacy system functions and the associated incremental introduction of the new system's capabilities into your operational environment.

4. Under what circumstances does it make the best sense for a state Child Support agency to consider this new approach versus other possible means of modernizing its Child Support system? Are there any characteristics of either a state’s IT system or its business processes that lend themselves more to this approach?

We believe the opportunity to leverage Low Code/COTS and incremental modernization strategies can be appropriate regardless of the architecture of the legacy system or the business process characteristics of a particular state. There may be, however, unique circumstances that make this approach even more attractive.

If a state has defined an Enterprise Architecture for its strategic systems, for example, there may be an opportunity to leverage software agreements or other components that have been selected as state standards. In many states, the Child Support systems are among the last still running on the mainframe. This reality may drive the decision to move to a “best of breed” approach that could be deployed to the Cloud.

In many states there are complex relationships between stakeholders, limited funding availability or political realities demanding low-risk approaches designed to deliver tangible results quickly. In particular, the relationships between key stakeholders can often provide challenges and impediments to effective modernization.

Typically, key players in the process are not only in separate divisions of a state department, in many cases they may be separately elected officials who have responsibility for essential components. This may include judges, prosecutors or court clerks. This complex environment requires collaboration and often designing an approach which results in smaller, incremental “wins” to maintain momentum and enhance the prospects for project success.

Finally, we strongly believe that the needs of these complex business process are best served by solutions that are built-for-purpose. The business of Child Support is unique and is best served by a platform that can support the complexities of case management rather than a generic approach of retrofitting a CRM or ERP platform designed to serve the commercial sector.

We believe Low Code/COTS, when combined with an incremental modernization strategy:

- Controls risk with a phased, yet holistic approach
- Achieves faster results with quick wins
- Takes advantage of the latest technology
- Can be configured based on State-specific priorities

5. **Generally speaking, what should a state expect on the following: project timeframe, project cost, time to rollout statewide?**

PROJECT TIMEFRAME

Generally speaking, the timeline for getting valuable new functionality deployed into an operational environment using a Low Code/COTS solution should typically be shorter than that of a transfer system or custom build scenario. As with most projects, a number of variables can impact the overall timeline. Among the areas to be considered include:

- The availability of agency subject matter experts – particularly those with certain levels of approval and/or decision-making authority. You want your team to be working together with your solution provider from the very beginning of the project to the end. Few things cause more project schedule slippage than delays in attaining key approvals and decisions.
- The readiness of your agency for the organizational change management (OCM) that is so essential to the success of a large modernization initiative. The implementation of the new technology generally becomes the (relatively) easy part. The impact to the people and the processes involved typically emerges as the most complex aspect of the project.
- The quality of fit and alignment of your chosen core Low Code/COTS products with the ultimate functionality that will be required and desired in the new system. A Fit/Gap Analysis is often undertaken as an early activity to (literally) find both the “fits” and the “gaps” in the product’s baseline functionality compared to that which will be needed. Such an analysis can serve as the baseline for the estimation of the level-of-effort that will be required as well as the associated timeline. Note that the true extent of the Low Code configurability provided by the core COTS product(s) being implemented can have a very real impact on the timeline and the associated cost.

In short, there is no simple, one-size fits-all answer to the timeframe question. Each modernization project will have its own unique set of circumstances that will inevitably dictate the realities of the actual timeline and ultimate project completion. However, generally speaking, the advantages of a product that supports configuration and avoids programming is an accelerator to the project timeline.

TIME TO ROLLOUT

Low Code/COTS solutions lend themselves particularly well to an incremental roll-out approach, where legacy functionality is retired and new functionality is introduced into the operational environment in logical, planned increments – and not as a single Big Bang years after the project starts. New, value-added capabilities are in the hands of the users sooner, “quick wins” can be achieved to facilitate early momentum and buy-in for the project, and a certain degree of improved risk management can be realized with the more controlled, manageably-sized release



of new capabilities to the field. The same beginning-to-end project timeline may be maintained, but with many of the benefits and return on investment gained much sooner.

PROJECT COST

Replacing or modernizing a Child Support system is indeed a major undertaking. The team and staff needed to perform the gap, build, test, and go live requires a real commitment from both the vendor as well as the State. Here are some factors to consider regarding the project cost.

Configure and Build

Project cost is driven largely by product licensing and the labor cost required to configure and build functionality based on the gap determined from the core product and the state's requirements. The cost will naturally vary depending on the level of requirements for your system. For example, if as part of your solution you want to include predictive analytics and a data warehouse then this would be an additional component which would increase the project cost.

Proven Platform

Low Code/COTS products have proven functionality built in and maintained by their product vendors. This is a cost savings to the state as product features and enhancements are rolled out by vendor. Using a proven platform eliminates many of the common issues as these have already been addressed and rolled out by the product vendor.

Total Cost of Ownership

For products that are highly configurable (Low Code/COTS), future changes and enhancements to your system can be completed more easily and quickly. Nobody likes to wait months to implement a change to a report or even longer for more significant functional changes. And yet that is so often the case with today's systems. That lead time also implies additional cost to keep the system maintained so that it best serves the changing needs of your program in your state. A more easily configured system saves time and money not only in its initial implementation, but for the life of the system - leading to a lower total cost of ownership.

6. The states don't want to again face the major system build and cost challenges once they have modernized. If they choose this approach, what is the continuous improvement model for the platform? Will the states benefit from the vendor efforts without major costs?

Most Low Code/COTS vendors - particularly those who offer their product in a cloud-based "Software-as-a Service" model - will provide the ability for states to reap the benefits of the vendors' ongoing efforts to enhance and refine the product on an ongoing basis, as part of a scheduled recurring refresh cycle. The states will typically have some power to control the changes or updates to be applied to their specific implementation – in particular, those changes that have the potential to impact their unique existing configurations of the system.

Earlier in this document, we described the concept of incremental, or Agile, implementation of systems. This approach lends itself very well not only to the initial deployment of baseline system functionality, but also to the ongoing incremental rollout of system updates and enhancements over time. A well planned and coordinated partnership between the State and its vendor is a key aspect of establishing and ensuring a successful shared roadmap for system evolution over time. Elements to be considered in such a roadmap may include the frequency and timing of ongoing maintenance releases, known plans for product and/or implementation-specific enhancements, a framework for responding to federal or state-mandated requirements changes, and a model for ensuring a controlled introduction of updated product functionality to the State's specific configurations.

Among the technical factors contributing to achievement of these cost savings with the implementation of a Low Code/COTS solution are:

- With Low Code/COTS, the data model and the system code are separated, reducing coupling and making upgrades to the system code easier;
- With Low Code/COTS, organizations can add new meta data-driven configurations to add new modules without requiring a new code deployment.

It should be noted that there is the likelihood of at least some cost to be associated with even the best approach and plan for keeping up with the enhancement and evolution of the Low Code/COTS platforms. The objective should be to realize the benefits of the continuously improving product while keeping the cost to do so in-line with the actual benefits of those enhancements. Among the keys to achieving this goal would be (1) the selection of a product whose base capabilities are well-suited with the agency's needs, and (2) a competent, high-quality initial implementation of the product that is aligned with that vendor's intentions and best practices. When compared to the alternative of updating and maintaining a large custom-built system, the ongoing cost savings of a Low Code/COTS approach should be quite significant over time.

7. **What are the most important things that a state should do to prepare for this approach?**

The State might consider creating a roadmap to lay out desired strategic objectives and related tactical milestones on a timeline. This roadmap would include the state's priorities and the sequence of functional areas to improve (and therefore the potential order for implementing new system capabilities to support and enable these improvements). This roadmap and its related analysis might also include things like:

- Identifying and recruiting internal “champions” for the modernization initiative. Ideally, this group of champions should include representatives from various levels within the “business” side of the agency, the supporting information technology group, procurement, and other groups who will be required to make the initiative a success. Start early in this development of your team of champions; they will be key to your success!
- Somewhat related to the cultivation of champions for the project is the readiness of your agency for the organizational change management (OCM) that will be so essential to the success of a large modernization initiative. As we've note earlier in this document, the implementation of the new technology generally becomes the (relatively) easy part. The impact to the people and the processes involved will emerge as the most complex aspect of the project.
- A big part of your roadmap will be shaped by determining whether or not there is an opportunity to implement the functionality of the new system incrementally - or are there operational/environmental/political drivers that make a single big bang approach the only option? A big bang roadmap looks quite a bit different than the more desirable incremental one.
- As you are doing in this questionnaire, seek to gather current information and counsel from both your colleagues in other states as well as from the vendor community. There is huge investment going on in Child Support solutions within the vendor community, with many players jumping in for the first time, as well as longtime solution providers (like us here at RedMane) investing in updating their offerings. Take the time to learn what your options are and take extra care to separate the reality from the hype.

In short, the handful of preparation activities we've suggested above obviously represent but a few areas for consideration as you get ready to move into the future with your Child Support system modernization efforts. While the Low Code/COTS software topic that is the focus of this questionnaire revolves around a fairly technical area within the system modernization spectrum, RedMane's real-world experience of the past twenty years has taught us that the “people” dimensions of such initiatives always have the biggest impact on your prospects for success!

8. How does this type of Child Support system fit with states who need to have an enterprise approach? Many of the platforms seem to be creating the same old silos on a new platform. Is it possible to have one casefile for each person/family across the systems (Child Support, SNAP, TANF, family services, etc.)?

Historically, state information management systems were built and deployed in discreet silos to match the specific requirements of each program area, e.g. SNAP, Child Support, Child Welfare, Medicaid, etc. This build and deploy-in-silos strategies were largely undertaken as a result of how these systems (and the programs themselves) are funded. Each Federal oversight agency has their own regulations with specific directives on information to be captured and reported back to them, and each Federal oversight agency has a distinct budget to be spent on systems that serve their own program area. The result has been program-specific IT system procurements and the siloed systems as a result. This was, and still is, much more of a policy issue than a technology issue.

The desire for an “enterprise approach” to match the whole person care model of supporting the needs of an individual holistically and in concert with their family members, is indeed how these systems should be built and deployed. No person in need of assistance deliberately segments his or her needs or asks for help by isolating health and safety needs from his or her economic security needs.

IT systems shouldn’t be designed or deployed this way either. The “enterprise approach” to system design and development is critically important to achieving the goals of our government support programs and ultimately for assisting those in need to gain independence.

The advent of Low Code/COTS technology is a great leap forward in the ability to quickly and affordably tailor IT systems to the specific needs of enterprise organizations. Further, the availability of Low Code/COTS solutions to provide off-the-shelf capabilities as pre-defined - but easily modifiable and extendable - makes these new systems more affordable and more quickly deployable than ever before. The evolution from “custom code” to “COTS” to “low code/COTS” has taken many years to achieve, but the benefits of these modern systems are clear when they are deployed for the purpose that they were designed to satisfy.

For example, deploying a Low Code/COTS case management system that was designed for a public sector context will provide lower overall total cost of ownership, a shorter time to implement, be more easily modifiable to fit evolving business needs, and be eminently expandable to meet the needs of the agency than one that is not aligned with public sector case management. On the other hand, a generic Low Code/COTS CRM adapted from a commercial sector sales management context may not deliver the same results as quickly and may take substantially greater configuration work to achieve agency objectives. In summary, a well-aligned COTS platform (i.e. built for managing people and cases) will support Child Support and the other programs in an integrated way very well. On the other hand, one that is not well-



aligned adds challenges for a Child Support system, and this is only accentuated when extending the platform across multiple programs requiring similar, real case management capabilities. The ease of configuration associated with Low Code/COTS may attract the pitfall identified in the question— “creating the same old silos on a new platform.” This outcome is a consequence of not doing the challenging pre-work associated with adoption of new enterprise systems. It is not an inherent quality or consequence of the technology itself.

To speed directly into system configuration and deployment without organizational change management, or failing to inventory and seek common ground in cross-agency policies - ultimately not taking enough time to review these inputs and generate a business re-design in alignment with “enterprise thinking” – will ultimately compromise the creation of an “enterprise approach” to the resulting system.

The Low Code/COTS systems absolutely make it possible to have a single case file for each person across the various program areas, with linkages and relationship networks defined by program area, with security to enforce view & edit privileges and report on cases holistically and by specific program area. **However, achieving the “enterprise approach” from a technology perspective requires agreement on creating an “enterprise approach” from a business perspective.** To proceed with configuration work on a new system prior to defining the standards and nomenclature and data elements that will be shared or restricted across the enterprise will result in the creation of the same silos on a new platform—which is a less than ideal outcome for these modernization projects.

There is an exciting opportunity to implement positive change in Child Support and other related public assistance programs with Low Code/COTS technology. Creating the “one casefile” that accurately represents the needs and service enrollments of each individual in care that facilitates the development of a holistic and coordinated approach to achieving the best possible outcome for each case, across the enterprise, is within reach—more quickly and more affordably than ever before.

9. What haven't we asked that we should have?

There are specific advantages and considerations associated with the selection of a Low Code/COTS platform for Child Support systems modernization. We would suggest the following additional questions and answers as relevant information for your consideration.

Suggested Question 10: Relative to long-term upgradeability, what are the risks or advantages of a Low Code/COTS solution for Child Support?

The ancient Greek philosopher Heraclitus is credited with the quote “change is the only constant in life.” It is important to keep this thought in mind when considering a path forward for critical infrastructure, in particular for enterprise information technology.

The needs of the business of Child Support, customers, staff, regulators, etc. are in a constant state of evolution. Deploying a new Child Support system into such a dynamic business environment necessitates a strategy based on incremental, modular, “quick win” deployments that are reflective of the changing needs of the business. The most critical enabler of such a strategy is the technology itself, as it is not possible to execute such a strategy with information systems that take months or years to update through custom coded development efforts.

The selection of a Low Code/COTS solution for Child Support can be a critical enabler of rapid configuration in alignment with the evolving needs of the business. But not *every* platform is the same. Certain Low Code/COTS platforms have Child Support specific functionality already built into the platform, while other platforms are more “vanilla” out of the box. While the latter option can still yield a successful project outcome, the deployments will likely take longer and be more costly to put into production due to the level of configuration required. Alternatively, platforms that provide real, off-the-shelf, pre-built configurations for the business of Child Support have a distinct advantage for state agencies and can serve as a true accelerator of progress towards achievement of modernized systems.

In addition to the ability to change quickly and affordably in alignment with evolving business needs, the selected Low Code/COTS platform vendor must be able to upgrade its core technology platform to stay abreast of the latest advancements in technology without damaging the *unique* configurations of any one customer. This upgrade path gets harder to maintain the more *customized* your system becomes—namely by extending the platform to meet certain needs of the business that were not originally anticipated by the platform vendor or that are peripheral to its core mission.

APPENDIX A

In our response to Question #1 above, we described the evolution of software systems – from the classic custom-built coding of legacy systems to the emergence of COTS – specifically, Low Code/COTS – in more recent times. Along with this evolution of the software itself, there has also been a real transformation in the approach – or methodology – used in the development of information systems.

In this appendix, we offer a brief overview of this evolution of system development methodology – with the objective of providing you a definition and some historical context for concepts such as “waterfall”, “incremental”, and “Agile”.

WATERFALL METHODOLOGY

The methodology that has been used in the development of many large, custom-built legacy systems is known as the “Waterfall” methodology – where a prescriptive series of project phases were performed in a fairly rigid, highly sequential fashion. At the highest level, these project phases included Requirements, Design, Development, Testing, Deployment, and then Maintenance. A visual depiction of the phases yields a cascading flow of the project phases – or, the “Waterfall”.

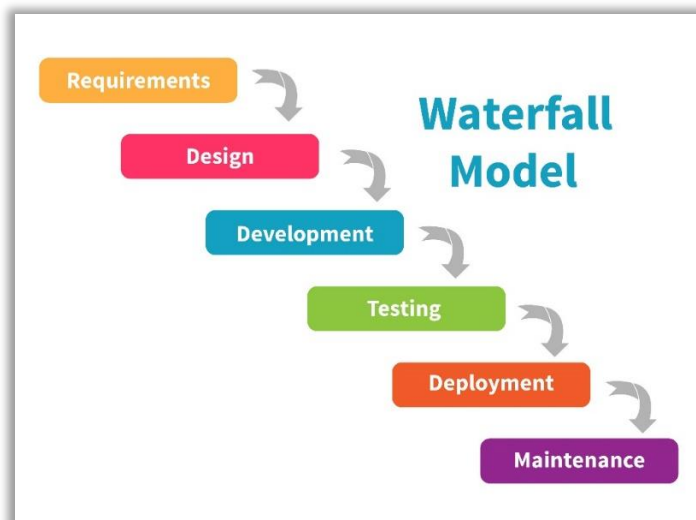


Figure 9: Waterfall Methodology

While the Waterfall approach provides a robust framework for building systems, there are some drawbacks, particularly when applied to very large projects. Among the problems encountered with this approach to large systems development is the challenge of keeping users and/or subject matter experts engaged, contributing, and aware throughout the project lifecycle. In the classic Waterfall project, users may be engaged upfront, during the Requirements definition

phase (during which **all** of the system’s (known) requirements are captured), but then they may not see the implementation of those requirements as part of a software system until many months or even years later, during Testing or Deployment activities – after the big army of programmers has gone off, interpreted the requirements on their own, and custom-built the system that in the end may or may not resemble and provide the capabilities actually desired and required by the users of the system.

Such long stretches of time between episodes of user engagement can be a major contributor to the misalignment of true needs and the ultimate shape and direction the system takes as the Waterfall phases progress. You may have heard this approach referred to as the “Big Bang” model, where all of the new system’s functionality is implemented in a single, typically huge, release into the production environment - with great shock and disruption to the people and processes of the affected organization – and often failing to deliver the desired results envisioned at the beginning of the project years earlier.

AGILE AND OTHER INCREMENTAL APPROACHES

In recent years, the concept of a more incremental approach to systems development has emerged as a more practical alternative to the Big Bang mentality of decades ago. The premise here is to achieve system modernization in a more manageable, risk-controlled fashion that allows for the gradual retirement of legacy system functions in a stepwise, incremental fashion.

The term “Agile” is often associated with such an incremental approach to systems development. Agile itself is a well-defined and documented methodology for iterative systems development, and in fact, in the nearly two decades that Agile has been practiced, a number of Agile variations and adaptations have emerged and put into practice.

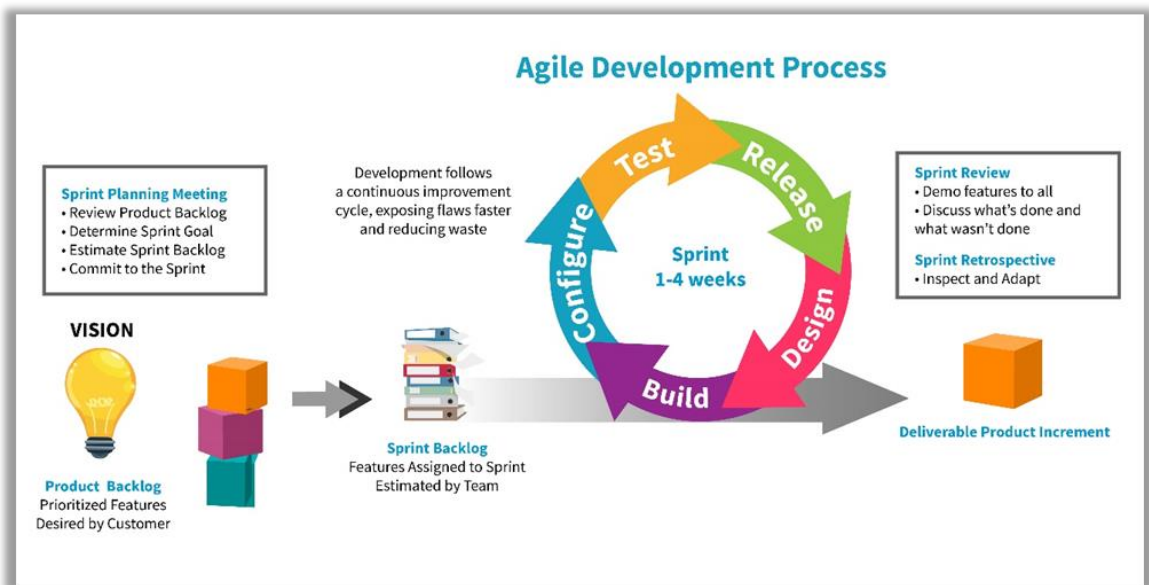


Figure 10: The fundamentals of an Agile system development process

For our purposes here, we don't need to get tangled up in theoretical or textbook debates around Agile adaptations and best practices, but the following principles can be thought of as key foundational aspects of Agile or incremental systems development:

- **Frequent Delivery:** As opposed to the years-long delivery cycles of yesteryear, we pursue frequent (measured in weeks or months, not years) implementation of working software. This quick turnaround approach facilitates faster, more relevant and impactful feedback – enabling better adjustment and course correction early and often. The concept of “build a little, test a little, learn a lot” is a good way to think of this. Figure 11 provides a visual representation of this concept.
- **Business and Developers working together:** This key premise calls for the client “business” side analysts and subject matter experts to work together - as part of an integrated team – with developers and technical staff, throughout the systems development lifecycle, not just at the beginning. Note that this does take a commitment of business-side staff to be available and involved throughout the process, but this commitment pays off with the cultivation of knowledgeable, highly engaged business-side team members who can serve in the important role of “champions” of the new system for the organization.
- **Welcome Change:** Change in requirements is inevitable and should be embraced, even relatively late in the process. Lessons are learned, discoveries are made, and ideas are generated at many points along the journey. The evolution of requirements is a sign that the team is getting closer to what the organization actually needs – a good thing.

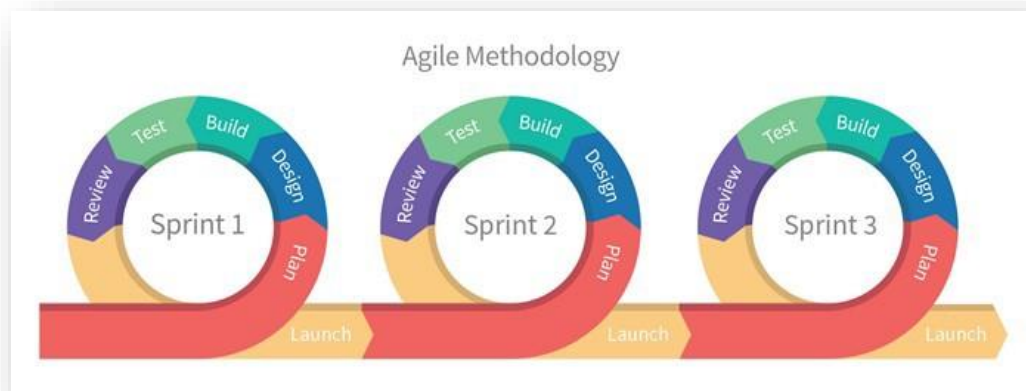


Figure 11: The iterative nature of an Agile system development process