



CRM Release Management Tips & Best Practices

Razwan Choudry

Microsoft CRM Consultant / Architect / Training Instructor

CRM Consultants UK



explore



engage



elevate

About Razwan

- Dynamics CRM Consultant since 2006
- Microsoft CRM Consultant Architect & Training Instructor available on ad hoc basis
- Author of 'Winning Strategies for CRM Success'
- Twitter @crmconsultants
- [My Free Solutions: XRM Toolbox – 'Bulk Merge Duplicates' Plugin / Autonumber / Geocoder & more on www.crmconsultants.co.uk](http://www.crmconsultants.co.uk)



Putting the CRM in SCRUM

- How can organisations successfully adopt the Agile approach with CRM implementations
- Factors to consider when Sprint Planning
- Risks and Impact of release cycles on overall CRM Success
- Managing parallel / concurrent crm development projects
- Understanding how Solutions work – Managed Vs Unmanaged
- Managing Development environments
- Automating the release process

Making Agile work for CRM

- Agile allows customers to deliver results quickly with incremental release cycles and sprints
- CRM is purchased as ready to use delivering Quick Wins / Fast Deployments
- Allows the customer to take ownership of their crm
- Empowers non developers to make customisations, but what are the drawbacks
- Agile may instil the mentality that defects can be fixed in later releases, potentially offloading the address of important design considerations such as performance
- Project Priorities may change, how can we ensure our solution release cycle will still work
- Agile doesn't work unless you have processes in place to support dependencies
- How can we improve release management & deployment Tasks with ALM automation
- Preventing solution errors & issues post deployment – Prevention is always better than cure even if agile allows us to 'Fix Later'

CRM is Forever

- New Requirements and Changes as Business Grows
 - New Customer Demands
 - New End User Demands
- Continuous CRM Updates and new Feature releases
 - New SDK Features
 - Deprecations of existing / old features
- Constantly evolving strategy to meet customer & business needs
- Eco System offerings disruptive to traditional methods
- Require repeatable processes to support changes

CRM Gremlins



CRM Gremlins

- Solution Import Failures
- Adding Dependant Objects to Solution Warning
- Missing Dependencies
- Deleting fields with same name / different type – SQL error during import
- Customise Form Button – Updates the Default Solution
- Shipping Core customisations from different environments / Object type code ordering could cause conflicts
- Solution Components with different Managed Properties/ States
- Unable to delete / remove solutions when dependencies are added
- Security Roles – Only root is included
- Workflow - needs to be owned by same person, otherwise you cant import
- Deployment tasks taking too long / Complicated
- Removing Unwanted components after importing solution

Post go live issues that should be considered

- Browser Issues / Compatibility mode
- Missing Config for Users / Email
- Security roles – access errors
- Outlook Client / Incompatible Add ons
- Emails not sending
- GUIDS are different? Breaking integrations / Workflows
- Licence types not working as expected
- Sitemap / Ribbon customisations – Consider Billing Admin
- Performance Issues often only reported post go live

Impact of Release Cycles on CRM Adoption

- Errors releasing solutions
- Preventing Upgrades
- Unsupported Customisations
- Limited Hotfixes and Support period
- User Adoption
- Increase in User Complaints
- Missing Deadlines
- Increase in project Costs

Importance of QA / UAT

- Often overlooked in CRM Project
- Given minimal priority / resources
- Issues are rushed close to release date
- How much resource & effort is put on Integration testing
- Performance should be considered during solution design
- Load testing / regression testing
- Could have wider implications / long term implications
- Once customisations have been released, its difficult to roll back for both Managed & Unmanaged Solutions
- Using TFS will help track and issues early on

Common Releases Issues

- Many resources often involved
- Difficult to manage and coordinate
- Missed out steps due to many manual tasks
- Out of Sync Development / UAT environments
- Rushed testing / Defects
- Unable to track who made changes
- Regular CRM Updates hard to keep upto date / SDK Changes

Customising Solutions

- Data Dictionary – share available fields and plan changes, helps prevent errors when creating objects
- Entity Schema design should be completed before every development project to plan for dependencies
- Separate / Isolate Development Environments to reduce dependencies
- Use a Core Solution to Maintain dependencies between Dev environments
- Use separate Env for Proof of concepts, create customisations from scratch to prevent Gremlins!
- Automate release process allows team to focus on their tasks

Risks

- Not adhering to a repeatable release process will have a direct impact on crm project success
- Project Complexity > complex development increases project risk
- Numbers of devs / size of teams
- Lack of Resources > ie QA?
- User Adoption > if IT staff struggle, Testing and UAT issues will have a direct impact on user adoption
- Large Infrastructure requirements
- Lack of Training

Problems with using Same Dev for All projects

- All Solutions required to begin as Unmanaged
- Soon as another solution is added will create dependencies on prior project
- Exporting Solution A after Solution B was Added will mean Solution A will automatically inherit any new dependencies created by Solution B
- This will break the Solution release process, even if we used a Master Solution for shared objects as solution will be exported with dependencies
- Single dev only suitable for simple and linear crm release cycle for single solutions or linear release projects
- Need to regularly backup versions of both Managed & Unmanaged from Development

Managing Dependencies

- New Dependencies are added to the earlier solutions every time a new solution is added with dependant components
- Exporting Solution A will include new Dependencies that were added by Solution A, so will not be able to import Solution A into Target Again
- Solutions C developed on environments without Solution B may result in Errors when creating relationships that already exist in Target, will not allow you to overwrite
- Occurs for both Managed or Unmanaged
- Solution layering should be considered during design / analysis stages not from development / release stage

Managing Dependencies

- All Solutions will Have Dependencies on Core objects
- The Release Order of Solution often determines how the Dependencies are managed
- When there is no clear release order, there should be effort to manage the core dependencies in a dedicated solution
- Using a Master / Core Solution will facilitate the release of multiple solutions
- Master / Core Solution should have Dedicated Development environment
- Project Specific Development environments need to be updated with Master / Core Solution at regular and controlled intervals
- Changes Made directly to production will contribute to Dependencies, cause issues with removing solutions. This when we should Consider Using Managed Components



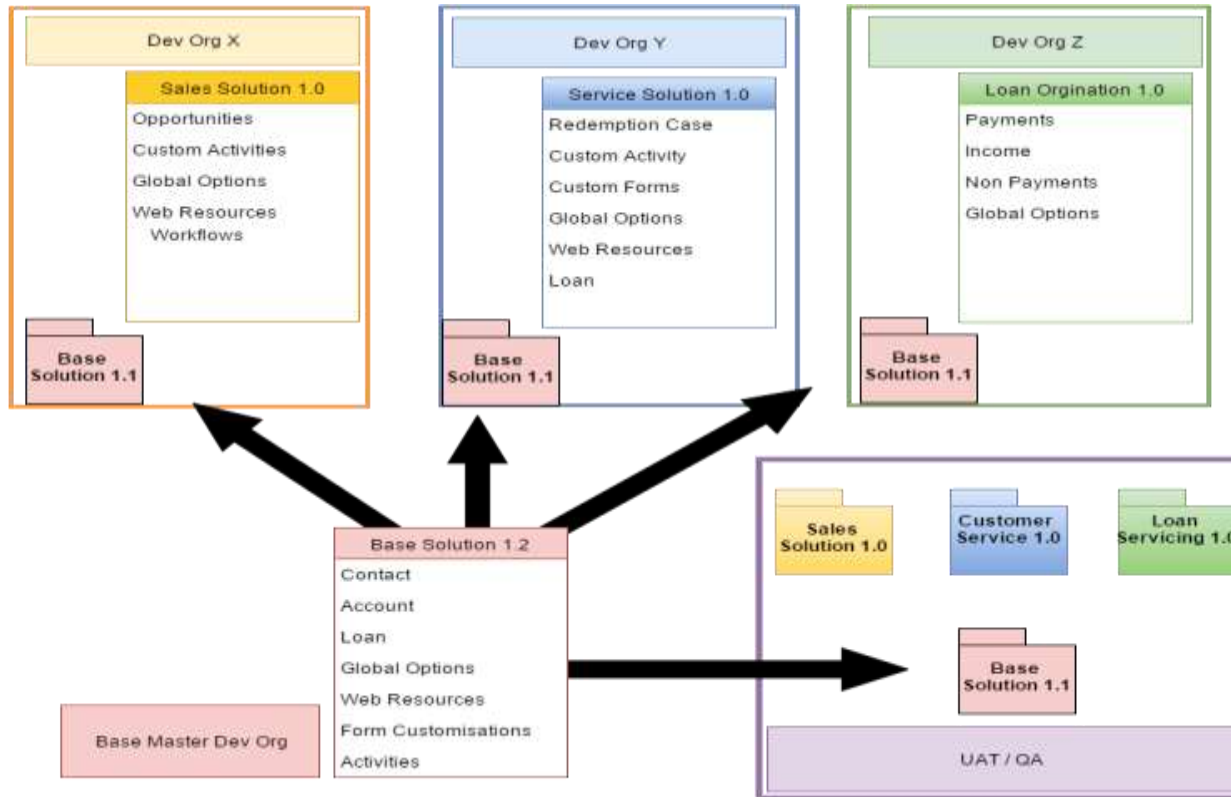
Creating Solutions

- How to structure your solutions for release?
- The recommended approach is to keep the number of solutions low to avoid complexity and ensure the solution customizations are independent from each other.
- Single Solution (Size limit)
 - 29.296 MB for crm online
 - 6 MB default for crm on premise
 - Difficult to apply hotfix for managed
- Component Based Solutions (size limit)
 - Merging , layering , ordering becomes more tricky
- Feature Based Solution (recommended)
- Recommended to isolate features into separate CRM solutions so that no merge conflicts & dependencies need to be resolved within CRM customizations.
- Incorporate a Core / Base Solution to Manage Dependencies when working in an environment with large development teams and fluxing release cycles

Managing Dependencies with a Core / Base Solution

- Export a New Solution from the Latest Development Environment, Containing only Common objects that may be used in other projects. Export this to your New Dev
- Now when you start development on your Project you will be able to consume existing components and add them into your solution rather than creating them if they already exist
- Any Changes made to the Core Solution Components by any projects need to be exported to all development environments using the Core Solution
- The Core Solution will need to be updated whenever another project updates components of the core solution, this can be done before

Managing Parallel Solution Deployment



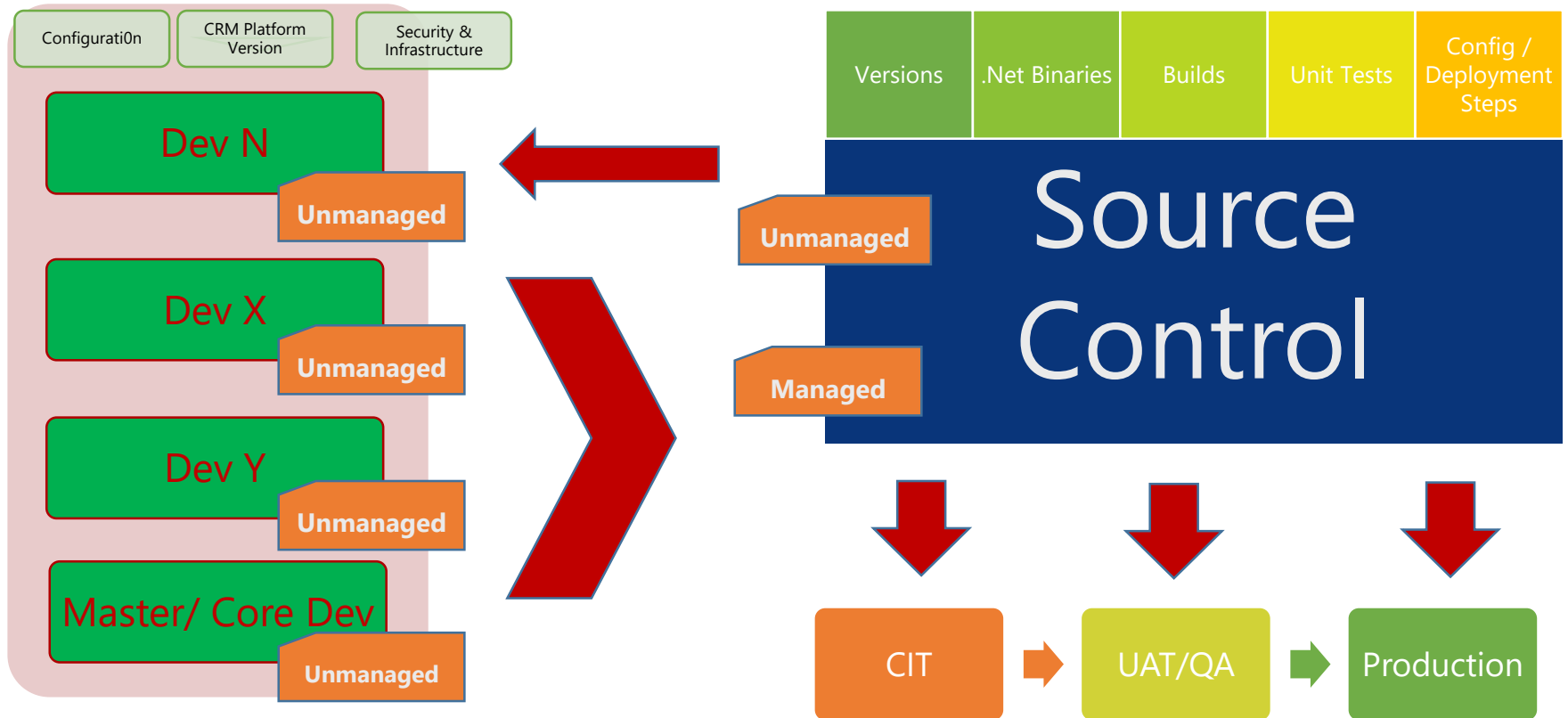
Solution Design

- Important to design Schema considering broader scope of future projects
- Include Existing and Related Projects in schema designs to show overall dependencies
- Data Dictionary will improve insight and collaboration
- Ensures the Objects are created with future projects in mind
 - Relationship Type
 - Reuse of Core Objects
- Help us be aware of dependencies
- Impact
 - Rework of Entity design
 - Break existing solutions

Managing Development Environments

- How can we ensure development environments and solutions are maintained with the correct version of solutions that will allow successful release into UAT & Production?
- Support development of Concurrent / Parallel CRM Solution Development
- Streamline the release process to facilitate Development, Testing and Go Live
- Ensure our Solutions are compatible with Rollups & Updates
- Problems with using the Same Development for all projects?
 - Dependencies
 - Conflicts
 - Testings
 - No isolation when trying to resolve issues

Managing Development Environments



Solutions to support Agile Release process

- ◀ The Solutions feature was introduced in CRM 2011, solutions are used to package your customisations and import/ export them to CRM deployments
- ◀ Empowers partners to use XRM & .Net framework to deliver ISV Solutions
- ◀ Import, Update and Maintain Customisation Components
- ◀ Publisher– Allows Multiple development teams and helps prevent conflicting customisations
 - ◀ Add Prefix to Entity and Field Attributes
 - ◀ Define Integer Range for Picklists/ Optionsets.
- ◀ Version Number – Manage Development / Releases / Updates / Hotfixes
- ◀ Compatibility to import from earlier CRM Versions, Option to export to specific target CRM version

Solutions Overview

- Solutions Customisation Components
- Settings
- Publishers
- Versioning
- Managed Vs Unmanaged
- Import & Export
- Creating
- Updating Solution Components
- Removing Solution Components
- Layers
- Merging

Solution Components

- Solution files can contain the following CRM system objects
- Can contain versions;
- can be exported for change management.
- Can be Managed and Unmanaged
- Can contain Global Settings, But not personal settings

Schema

- Entities
- Fields
- 1:N / N:N Relationships
- Option Sets

Templates

- Email
- Contract
- KB Article
- Mail merge

Processes / Steps / Code

- Workflows
- Dialogs
- Business Process Flows
- Business Rules
- Actions
- Web Resources
- Plug-in Assemblies
- SDK Message Processing Steps

User Interface

- Forms
- Views
- Charts
- Dashboards
- SiteMap
- Ribbons

Miscellaneous

- Security Roles
- Field Security Profiles
- Connection Roles
- Reports
- Solution Publisher
- System Settings

Whats not included in a solution?

- Business Units
 - Users
 - Teams
 - Queues
 - Goals
 - Subjects
 - Product Catalogue
 - Personal Views / Dashboards
 - Personal User Settings
-
- Customer Data

Working with solution files

- Solution file consists of;
 - Customizations.xml
 - Solution.xml
 - Workflows
 - Web resources (built by the web resource file projects)
 - Plug-in assemblies
- The customizations.xml file is exported in the solution zip. Certain portions of the customizations.xml file can be edited manually. SDK\Schemas provides XSD to help you make supported modifications
- Editing a managed solution file is not supported
- Not all elements of the customizations.xml file can be edited. However all these can be done GUI tools
- Extracting the Solution file is useful when comparing differences

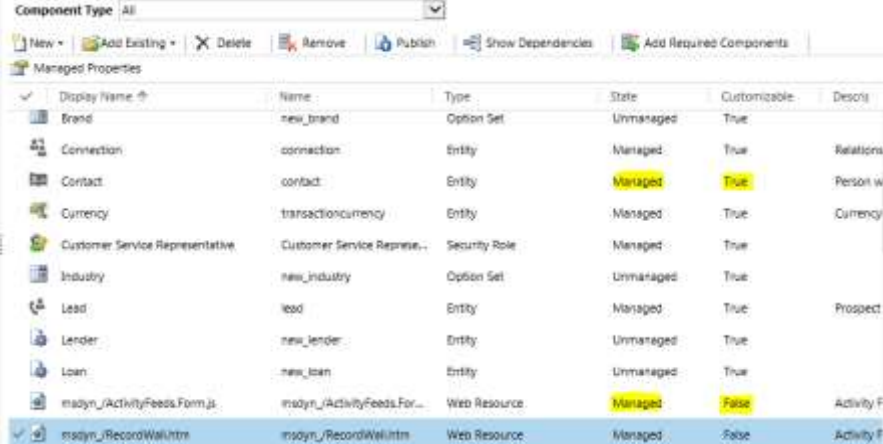
Layers

➤ To Understand Solution Truly, you have to think in Layers

- System Layer – Default Solution
- Unmanaged layer
- Manager Layer – Managed Components

Component States

- Why are System Components in an Unmanaged Solution have managed state by default?
- Keeps System or Business Entities as Customisable as these entities are used repeatedly for many other customisations projects
- Managed properties only become effective after their associated components are installed as part of a managed solution.
- The value or the IsCustomizable property, the type of entity, and the type of solution will have an impact on the behaviour of an entity
- Can be used use to prevent Customers from Deleting/ Editing System Objects or Breaking Core Functionality after being installed on production



Display Name	Name	Type	State	Customizable	Description
Brand	new_brand	Option Set	Unmanaged	True	
Connection	connection	Entity	Managed	True	Relations
Contact	contact	Entity	Managed	True	Person w
Currency	transactioncurrency	Entity	Managed	True	Currency
Customer Service Representative	Customer Service Represe...	Security Role	Managed	True	
Industry	new_industry	Option Set	Unmanaged	True	
Lead	lead	Entity	Managed	True	Prospect
Lender	new_lender	Entity	Unmanaged	True	
Loan	new_loan	Entity	Unmanaged	True	
msdyn_ActivityFeedsForm.js	msdyn_ActivityFeeds.For...	Web Resource	Managed	False	Activity F
msdyn_RecordWall.htm	msdyn_RecordWall.htm	Web Resource	Managed	False	Activity F

Unmanaged Solutions

- All Solutions begin as Unmanaged
- Unmanaged Solution customisation are Made at the Unmanaged Layer which is also part of the Default Solution
- Allows us to make customisations, register plugin steps etc, before we can export as Managed
- Deleting the Unmanaged Solution only deletes the reference to the solution, not the customisation components
- Requires manually deleting the customisation components from either the solution or Default Solution

Managed Solutions

- Managed Solutions need to be exported as Managed from an Unmanaged Solution
- The whole point of Managed is locking down the Component states so they cannot be edited
- This secures the solution in the Target / Production so it keeps the solution feature working and prevents end users from breaking it. Therefore it is maintainable.
- Managed Solutions are installed at the Managed Layer
- Deleting the Managed Solution will remove all its customisations as well as data contained
- Any Managed components that are customisable will be done at the Unmanaged Layer

Managed Vs Unmanaged Solutions

- CRM Supports using a mixture of both and allowing multiple solutions to easily apply bug fixes or updates on the environment.
- The downside is there is no standard mechanism of removing fields from entities, webresources etc. Could create many redundant objects.
- The ALM for Microsoft CRM officially states that Unmanaged solution is only used for development and Managed is released down stream to production. (Ideal for ISVs and works for incidental development or where development is following a fixed release calendar, but what about for Agile scenarios where the Customer is responsible for their own deployments?)
- Once you release as Managed you cannot covert back to Unmanaged, however an Unmanaged solution can be converted to Managed
- However Managed Solutions may interfere with the Agile Release Process where the solution is being developed, tested and used at the same time. If a managed solution is deployed to Test, QA and Production, If a blocking issue occurs in the production environment it will take longer to resolve. It may be better to use Unmanaged so you can take a snapshot of production, replicate & resolve, update unmanaged solution and release to Production.

Updating Solutions, Layers & Limitations

- Solution Component needs to belong to either a Managed or Unmanaged Component, Cannot be a Mixture of both!
- Components in the unmanaged solution layer cannot be updated by a managed solution.
- Cannot change the attributes of unmanaged solution components so that it is in a managed solution layer, so it can be edited or changed by a managed solution.
- Manually deleting the components that the unmanaged solution installed, you can import that solution again as managed and then install an update for it with a managed solution. However you will lose your data, will need to export the data prior to deletion and then re-import it when the components are installed as managed

The Solution Publisher

- ▶ Publisher– Allows Multiple development teams and helps prevent conflicting customisations
 - ▶ Add Prefix to Entity and Field Attributes
 - ▶ Define Integer Range for Picklists/ Optionsets.
- ▶ **The solution publisher for a managed solution plays an important factor when releasing an update to your managed solution. Using the same solution publisher you can create a new managed solution to update a managed solution you previously released.**
- ▶ **Components in managed layers will be owned by the solution publisher.**
- ▶ Publisher owns the component, not the solution.
- ▶ Components with same name and publisher will be considered the same thing.
- ▶ Removing a solution does not remove a component when it is referenced by another solution using the same shared publisher.
- ▶ Default Publisher created for each environment with same default new_prefix and Integer range for optionsets
- ▶ Important when creating development environments

Updating Solutions Considerations

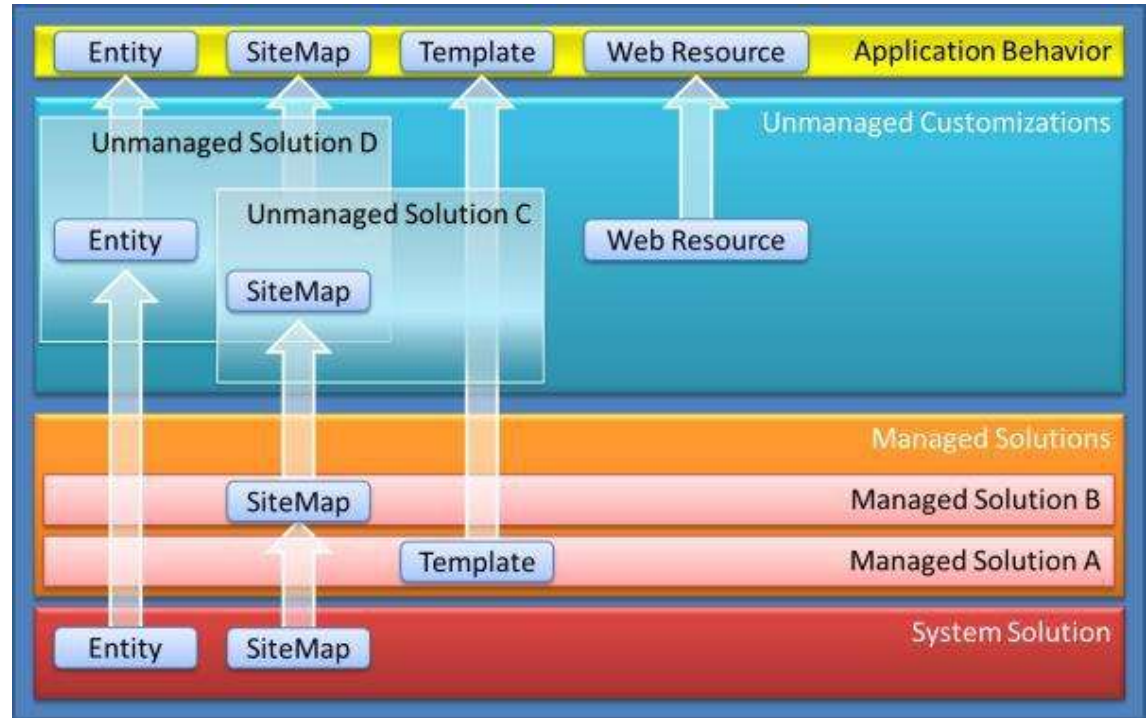
- Importance of Publisher And Solution Name when Updating Solutions
- **Maintain customizations (recommended)** - This option maintains any unmanaged customizations performed on components, however some of the updates included in this solution may not take effect.
- **Overwrite customizations** - Overwrites any unmanaged customizations previously performed on components included in this solution. All updates included in this solution will take effect.
- Using the **Overwrite customizations** option when investigating customizations conflict issues. They can export their unmanaged solutions before so that they can re-apply if required.
- "Creating a new solution with a different name and same publisher, will appear to work until the next version of the solution is deployed. Selecting the option to import without overwriting will import the new version of the solution within a layer directly above the previous version of the core solution. This will be beneath the solution containing the hotfix, which therefore may result in configuration and customization changes not being surfaced."

Applying Solution Updates / Hotfixes

- Solution must maintain the same solution publisher and the same solution name
- The Installed Date remains the Same when updating a solution, which can be misleading
- Updating the Solution does not cause loss of data
- Maintain the version number by overwriting the existing solution (only include the affected components)
 - Using the recommended approach to preserve customizations when importing managed solutions, maintaining the version number will automatically replace any components within the solution being imported
 - Lack of ability to identify that a hotfix has been applied. Consider adding a note
- Or with an incremented version number of the existing solution and create a new layer
 - incrementing the version number will create a new managed layer for the solution above the old layer.
 - can become misleading, as it suggests that the hotfix can be uninstalled or deleted which is not necessarily true due to the additive nature of the platform.
- Creating a new solution with a different name and same publisher, will appear to work until the next version of the solution is deployed. selecting the option to import without overwriting will import the new version of the core solution within a layer directly above the previous version of the core solution. This will be beneath the solution containing the hotfix, which therefore may result in configuration and customization changes not being surfaced.
- Creating a New Solution with just the updates will allow you to rollback, but will add an additional solution into your organisation, which could potentially clutter

Merging

- Merging is how CRM will Virtually Merge the Behaviour of the multiple solutions according to the following rules
- Allows combination of many Managed and Unmanaged solutions
- Order of Solution Imports may effect behaviour



Deleting / Removing Solutions

- Managed Solutions will be removed including data
- Deleting the Unmanaged Solution will only delete the Solution Container, However all Customised Components will Remain. Will have to manually delete any unmanaged components from the Default Customisations
- Deleting a Managed Solution will Remove all the Customised Components included in the Solution
- Dependencies are known to cause Problems preventing to remove Managed Solutions
- Some customers have resorted to Unsupported Techniques when unable to uninstall Managed Solutions

Removing Customisations from Managed Solutions

- Solutions are 'Additive', so to Update a Managed Solution by removing a customisation a Temporary solution is required;
- Create a temporary managed solution for the customisations you want to keep using the same publisher, but with a Unique Name. This is so it contains the customised components at a separate managed layer.
 - xrm toolbox will be able to copy all the components for you , eliminating risk of missing any components
- Import it on top of your existing managed solution
- Delete the older managed solution
 - This deletes the unwanted managed components while the "holding" solution prevents the wanted customisations from being deleted
 - During the delete process, the CRM server finds anything that's from the same publisher and effectively says "dont delete because another solution from the same publisher claims you too."
- Import the new version of the managed solution
- Delete "holding" managed solution
 - The contents are the same, but in two managed layers. You delete the holding solution after the new version of the solution, While you could eliminate the last two steps, you would have to accept the change in UniqueName of the new solution what's really a new version of an existing solution. This will allow you to keep your Managed customisations without the data and customisations you want to keep

To prevent getting in this scenario its always worth performing a good design that is future proof, IE using Lookups instead of Picklists, Name Conventions etc

- Note :Editing a managed solution XML file is not supported.

Unmanaged Solutions in Production?

- Is it wrong to use unmanaged solutions in production?
- Depends on how the solution is being delivered. ie internally or externally
- Risks associated to how well IT infrastructure / TFS / Version control is maintained
- If the business has procedures in place to disable direct customisation to live / security
- Unmanaged Solutions are easier to maintain for customers
- Allows to snapshot production to resolve a blocking issue
- Merging behaviour for updates is simpler, less complex than managed
- Ensures customer business customisations & data are always part of the default solution
- Less risk of losing customisations as more difficult to remove

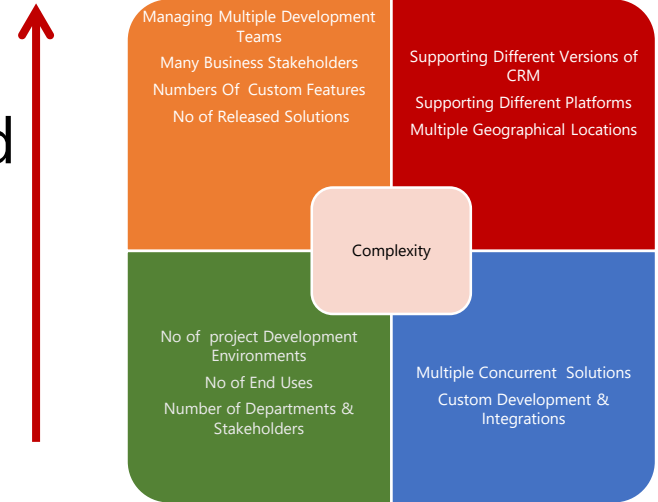
Types of Managed CRM Deployments

ISV Provider - Managed

MS Partner – Managed / Unmanaged

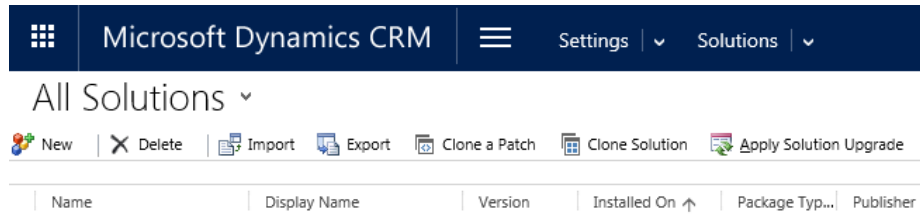
Customer - Managed / Unmanaged

- IT Department (In House)
- Several IT Departments for a Global Organisation



Solution Patching – CRM 2016

- Previously when you add an entity customisation to a solution and export the solution, all of its related assets would also be exported including attributes, forms, views, relationships etc. Exporting all objects means that you can unintentionally modify objects on the target deployment, or carry over unintended dependencies
- Now with CRM 2016 we can release solution patches which only contain the updated components
- This helps to simplify the release process for updating existing solutions as we no longer to to carry across existing components
- You can roll up all the patches into a new solution version to replace the original solution that the patches were created from.



The screenshot shows the Microsoft Dynamics CRM interface. At the top, there is a dark blue header with the Microsoft Dynamics CRM logo, a hamburger menu icon, and navigation links for 'Settings' and 'Solutions'. Below the header, the text 'All Solutions' is displayed with a dropdown arrow. A toolbar contains several action buttons: 'New', 'Delete', 'Import', 'Export', 'Clone a Patch', 'Clone Solution', and 'Apply Solution Upgrade'. Below the toolbar, a table header is visible with columns for 'Name', 'Display Name', 'Version', 'Installed On' (with an upward arrow), 'Package Typ...', and 'Publisher'.

Solution Patching – CRM 2016

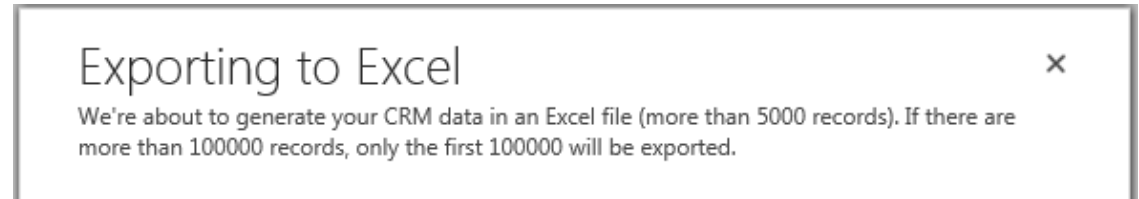
- ▶ CRM 2016 introduces following new features;
 - ▶ Clone a Patch
 - ▶ Clone a Solution
 - ▶ Apply Solution Upgrade -
- ▶ Allows to Isolate exact changes to customisations and improve Deployment Steps – Patches Will help improve the release process
- ▶ No longer automatically prompts to add related objects

CRM Deployment Tools

- Configuration Migration Tool
- Solution Packager
- Package Deployer
- ALM Toolkit
- TFS
- Surestep

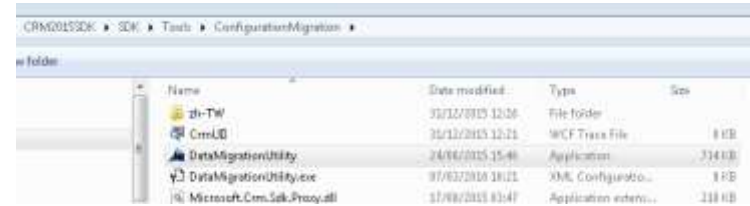
CRM Data Import Wizard

- Manually Import Excel File
- Create Mappings File
- Limited to first 100000 records)10,000 for pre crm 2015



CRM Configuration Migration Tool

➤ Included in the SDK



➤ Create Data Schema and XML

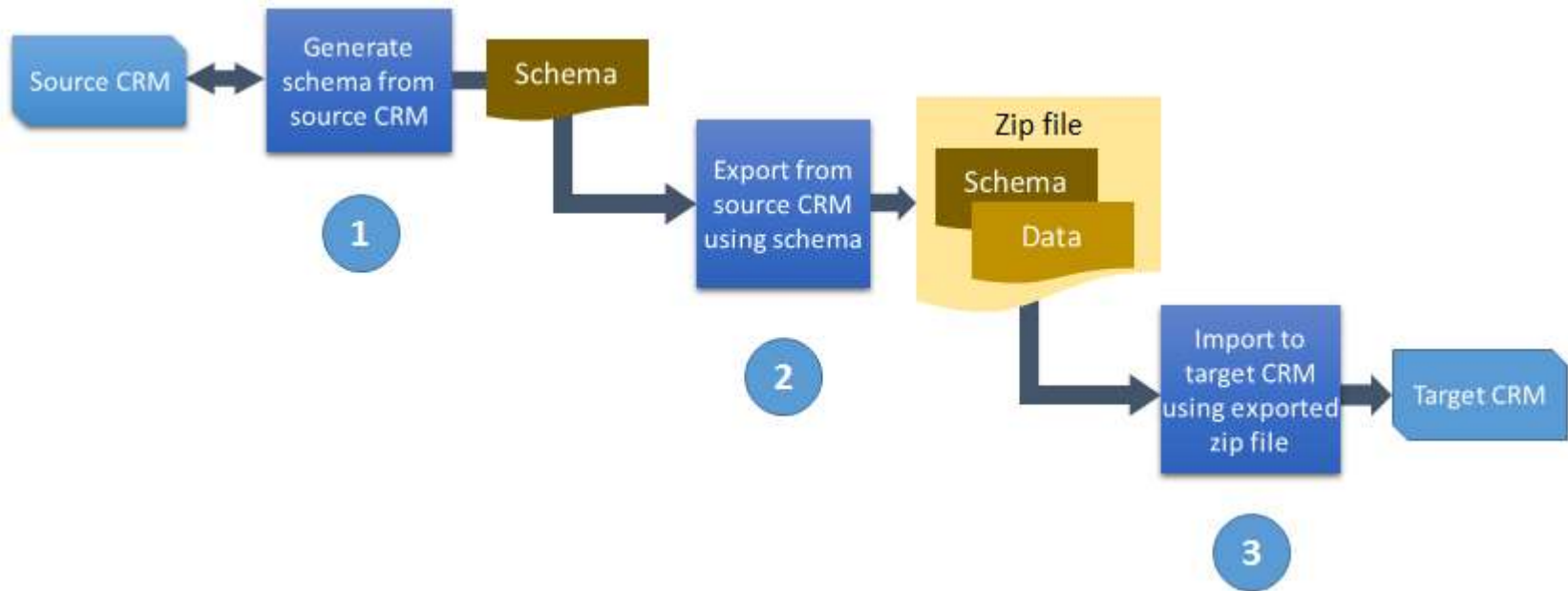
➤ Keeps Same GUID as Source, useful for deployment and Testing

➤ Great to use when there are complex relationships involved

➤ Limited to config data, Struggles with large Data Sets

➤ Doesn't update User Security Roles?

CRM Configuration Migration Tool



Creating Development Environments

- ▶ Snapshots of VM
 - ▶ Infrastructure requirements / resources available
 - ▶ Reliability if in correct state
- ▶ CRM Online Sandbox Copy function
 - ▶ Limited to copying online running sandboxes
 - ▶ Cost?
- ▶ Database Restores
 - ▶ CRM Environment may be different / out of sync
- ▶ Solution Files from Release Folder / Source Control
 - ▶ Replicates Release Process Steps
 - ▶ No requirement for CRM Organisation to be running
 - ▶ Config Data can be packaged separately

Managing Development Environments

- Development Environments should be isolated from UAT / Production
- Should be Disposable
- Should be created easily from Source Control / Snapshots
- Dedicated Environment for Each Project to support release process / Reduce Dependencies
- For On Prem Development , consider AD and Infrastructure differences ,
- Should include items not included in solution file
 - Config
 - Service Accounts / Licence / Privileges
 - Custom integrations

Versioning and Release Management

- Solution deployment processes need to be repeatable and predictable. Using development and version control techniques will support and simplify the approaches
- Even with Solution packaging and layering of specific components the process of managing parallel development and tracking customization changes can still be a challenge on enterprise Dynamics CRM projects
- CRM solution content is stored and versioned in a single binary solution zip file

Automating the repeatable release processes

- Key Factor for CRM Success!
- Support Agile Method
- Multiple concurrent crm projects & development teams
- Utilising Source Control
- Managing Development Environments
- Config Data
- Automate Release Process
- Collaboration
- Controlling Risks
- Reduce time spend on managing devs and releases
- Increase ROI

Benefits of TFS / Version Control

- Source Control for all copies of Solution files both Managed & Unmanaged
- Able to use these files to rebuild perishable development environments ad hoc
- Solution files are compressed
- Using Unpacked Solution files can be useful for tracking changes and comparing solution files

Version Control

- ▶ Storing uncompressed solution files enables the following individual CRM source elements to be independently version controlled and tracked:
 - ▶ Customizations.xml
 - ▶ Solution.xml
 - ▶ Workflows
 - ▶ Web resources (built by the web resource file projects)
 - ▶ Plug-in assemblies (built by .Net projects)
- ▶ All other elements in the customization and solution XML require manual creation or merge into one single source file by a developer as necessary

Tools

- ◀ Solution Packager:
 - ▶ Unpacking the solution xml files to customization elements
 - ▶ Packing the customization elements to a single solution package
 - ▶ Supporting managed and unmanaged solutions
 - ▶ TFS automatic build process integration
 - ▶ Pluggable framework to add processing steps to specific customization elements
- ◀ GUI editors
 - ▶ SiteMap editor / XRM Toolbox
 - ▶ Ribbon editor – Scott Durrow

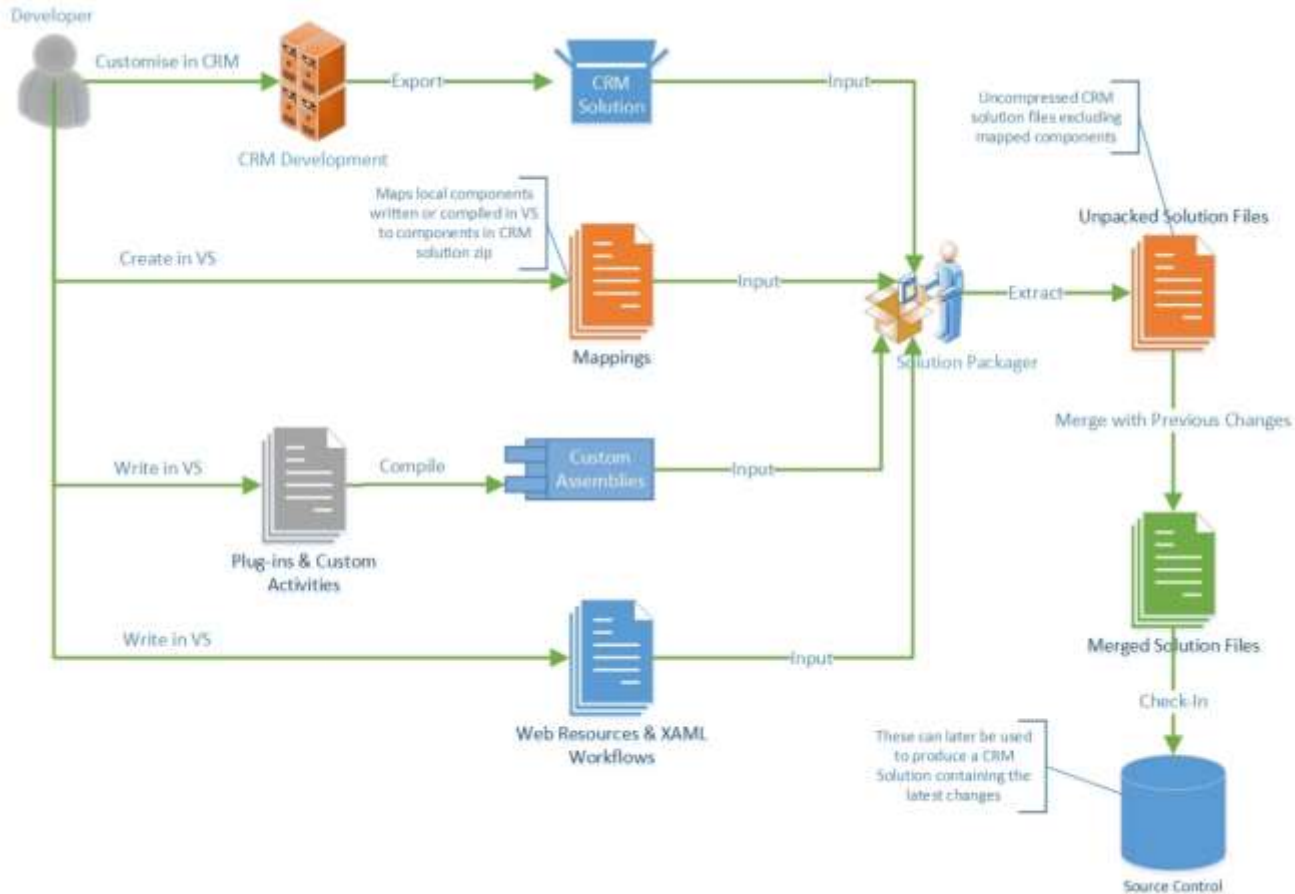
TFS Requirements management Build management Version control Bug tracking Test management Development reporting Team development, team management Source control and branching Development and code policies Localization QA process support Integrated Development Environment (VS)

Solution Packager

- Allows to overcome solution restrictions enabling to move customisation components between two environments
 - CRM solution development in isolated dev environments
 - CRM project with one or more parallel streams of work where each stream has their own environment and different release schedule but working on the same CRM Solution
- CRM Solution containing changes might potentially override customizations that are already in there. You might end up re-doing some of work manually in two places.
- Difficult to implement specific features such as changes to data model, workflows, user interface and others.

- Pack / Unpack Solution
- Mappings
- Merge

Solution Packager



Solution Packager

- Currently Bug For binary file comparisons, it has the wrong logic. Files that have not changed are treated as files that have changed, and vice versa.
- WorkAround: Use the PowerShell Cmdlet to delete the binary files before unpacking the solution. The Example below removes the Xaml files from the workflow folder.
- `Get-ChildItem -Path ...Solution\Workflows -Include *.xaml -File -Recurse | foreach { $_.Delete() }`
- when you pack the solutions files, you can use the mapping option to take the files from the source location.

ALM Toolkit / Release Automation

- ▶ Allows the use of Powershell scripts to manage the following;
 - ▶ Solution files
 - ▶ Config items
 - ▶ Deployment Tasks
 - ▶ Server Components / Tasks
- ▶ Reduces the need for all Infrastructure resources to have to be available every time there is a release or development server requirements
- ▶ Requires a central deployment manager / controller

ALM Toolkit is now Free!

- Now Free to download for both Developer / Commercial Use!
- Automate change management for Microsoft Dynamics CRM projects using TFS. The ALM Toolkit empowers project teams to apply agile methodologies to manage change, achieve developer isolation, and deploy solutions to multiple environments.
- Includes the following components:
 - CRM Solution Management PowerShell Scriptlets
 - CRM Data Manipulation PowerShell Scriptlets
 - CRM Data Copy Utility
 - Sample scripts for ALM environments
- <https://community.adxstudio.com/products/adxstudio-alm-toolkit/download/>

Summary

- ⚡ Sprint Planning should consider solution dependencies, performance factors to reduce risks
- ⚡ Control Solution Dependencies in a core / base solution to support parallel release management to support concurrent projects
- ⚡ Using collaboration such as Data dictionary will help prevent customisation errors/conflicts during dev
- ⚡ Address Risks & Performance considerations during solution design to avoid issues being missed in UAT / QA
- ⚡ Agile may delay underlying issues from being resolved, always best to factor in risks during design to prevent occurrence
- ⚡ Use Tools such as Configuration Manager to migrate Config Items not included in solution
- ⚡ Solutions are Additive, Removing Customisations already deployed is not so straight forward
- ⚡ Managed Vs Unmanaged - The way we choose to export our solution may involve additional complexities such as solution layering and merge behaviours
- ⚡ Use Source control such as TFS & Package Deployer for Regular backup streams for all solutions
- ⚡ TFS / Source Control useful for Rebuilding Development environments as well as Collaboration, Comparing & Tracking Solution changes
- ⚡ Using ALM toolkit can significantly improve release management and save resources
- ⚡ ALM toolkit is now free to use, Reduce time spent managing development environments and deployments
- ⚡ May take time to implement at first but will save time & resources over long term, "CRM is Forever"

Resources & References

- ▶ ALM for Microsoft Dynamics CRM 2011: CRM Solution Lifecycle Management <https://www.microsoft.com/en-gb/download/details.aspx?id=39044>
- ▶ ALM Toolkit <https://community.adxstudio.com/products/adxstudio-alm-toolkit/download/>
- ▶ Microsoft Dynamics CRM Software Development Kit (SDK) for CRM Online and on-premises



Any Questions?



explore



engage



elevate



Please fill in your feedback forms for this session!

Razwan Choudry

CRM Consultant / Architect / Trainer

Tweet me @crmconsultants if you have any questions



explore



engage



elevate