THE INTEGRATION RELATIONSHIP DIAGRAMMING METHOD (IRDM)

A Four-Step Framework for Structuring Complex Project Controls Without Breaking the WBS

Abstract

Conflicting definitions and inconsistent application of foundational structures - such as the Work Breakdown Structure (WBS), Organizational Breakdown Structure (OBS), control accounts, and work packages - continue to undermine the effectiveness of project control systems, even in technically compliant environments. While guidance documents such as MIL-STD-881F, EIA-748D, and the NDIA EIA-748 Intent Guide each prescribe sound principles, they diverge on how structure should support visibility, integration, and reporting. The result is widespread reliance on hybrid WBS models that collapse under real-world program demands.

This paper introduces the Integration Relationship Diagramming Method (IRDM) - a four-step framework for resolving structural conflicts and designing scalable, compliant control systems. Through a step-by-step comparison, the article demonstrates how IRDM preserves control account integrity, maintains a clean WBS, and delegates integration complexity to the work package level using structured metadata and naming conventions. The method is grounded in field experience from large-scale government programs and is aligned with EVMS best practices. IRDM enables multidimensional traceability, reduces process overhead, and restores structure as a source of clarity and control.

Jeffrey Christoph
jeff@transformativems.com
www.transformativems.com

Contents

Executive Summary

Modern project control systems are under pressure from every direction. Contracts are more complex. Funding is fragmented. Reporting requirements are multi-dimensional. Yet the structural foundations most programs rely on - particularly the Work Breakdown Structure (WBS) and control account hierarchy - have remained essentially unchanged. As a result, project teams are being asked to manage growing complexity with tools and frameworks that can't scale to meet the challenge.

This article begins by demonstrating how that failure plays out in practice. Using a simplified example, we show how a traditional hybrid WBS - where contract elements, phases, and reporting requirements are embedded into the structure - leads to bloated control accounts, obscured accountability, and fragile reporting logic. From there, we step back to define the structural elements at the heart of the problem: the WBS, OBS, control accounts, and work packages. Only by aligning definitions and clarifying intent can we resolve the contradictions across established guidance and standards such as MIL-STD-881F, EIA-748D, the NDIA Intent Guide, and AACEI Total Cost Management.

With that foundation established, we introduce the **Integration Relationship Diagramming Method (IRDM)** - a four-step process developed in response to a highly complex DoD program with over 40 CLINs, overlapping funding lines, and over 1,000 work packages. Rather than force visibility into a hierarchy, IRDM uses metadata, structured naming, and defined relationships to manage complexity without inflating structure. It keeps control accounts focused on control; work packages focused on execution, and integration aligned to purpose - not format.

This paper doesn't just propose a new method - it demonstrates it in practice, one step at a time. The result is a more scalable, traceable, and resilient system - without sacrificing compliance or control.

Before we can evaluate the strengths of a new approach, we need to understand exactly where the traditional model breaks down. The following example illustrates how a seemingly straightforward WBS - when forced to absorb programmatic visibility like CLINs and phases - leads to bloated control accounts, structural fragility, and management ambiguity. This is where most programs begin, and too often, where they get stuck.

How to Use This Document

This document is intended to serve multiple audiences involved in structuring, executing, or overseeing project control systems:

- **Project Controls Professionals**
  For schedulers, EVM analysts, and control account managers, this paper provides a detailed method for designing scalable structures, applying metadata, and using naming conventions to support integration and compliance.
- **Program Managers and Technical Leaders**
  For PMs and IPT leads, this document explains the structural consequences of legacy approaches and demonstrates how control can be maintained without overbuilding or losing visibility.
- **Executives and Oversight Stakeholders**
  For executives, DCMA reviewers, or governance bodies, this paper illustrates how structural clarity and traceability reduce risk, improve responsiveness, and align with compliance expectations.

Each section builds on the last - starting with a failure pattern, then introducing a practical solution and showing it in action. Use the detailed examples, figures, and summary tables to apply the method or evaluate your current program structures.

Program Organization with a Hybrid WBS Approach

To understand why structure matters, we'll begin with a simplified example. Using a generic product-oriented WBS and a representative OBS, we'll first apply a traditional hybrid approach that tries to layer contractual and programmatic visibility directly into the hierarchy.

Control accounts are established at the intersection of the Work Breakdown Structure (WBS) and the Organizational Breakdown Structure (OBS), representing the point where defined scope and assigned responsibility align - this is the foundation of earned value planning and measurement across all primary EVMS guidance documents, including EIA-748D, the NDIA Intent Guide, and the DoD EVMIG. The Control Account assignment image from the NDIA 748-D Intent Guide illustrates this principle clearly.

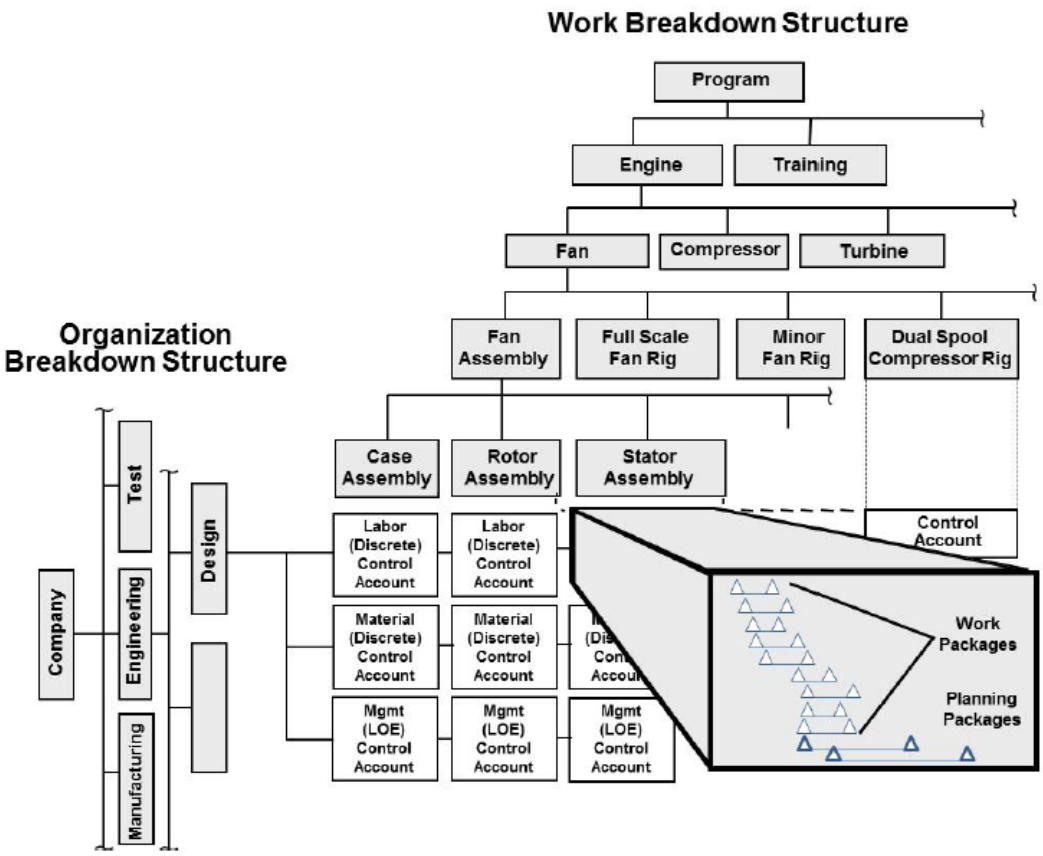


Figure 1-Establishing Control Accounts from NDIA EVMS Intent Guide

We'll start our hypothetical example using this guidance here. Shaded green boxes show the intersections where we define our initial control accounts. We extend the WBS coding down one level using a hybrid WBS model.

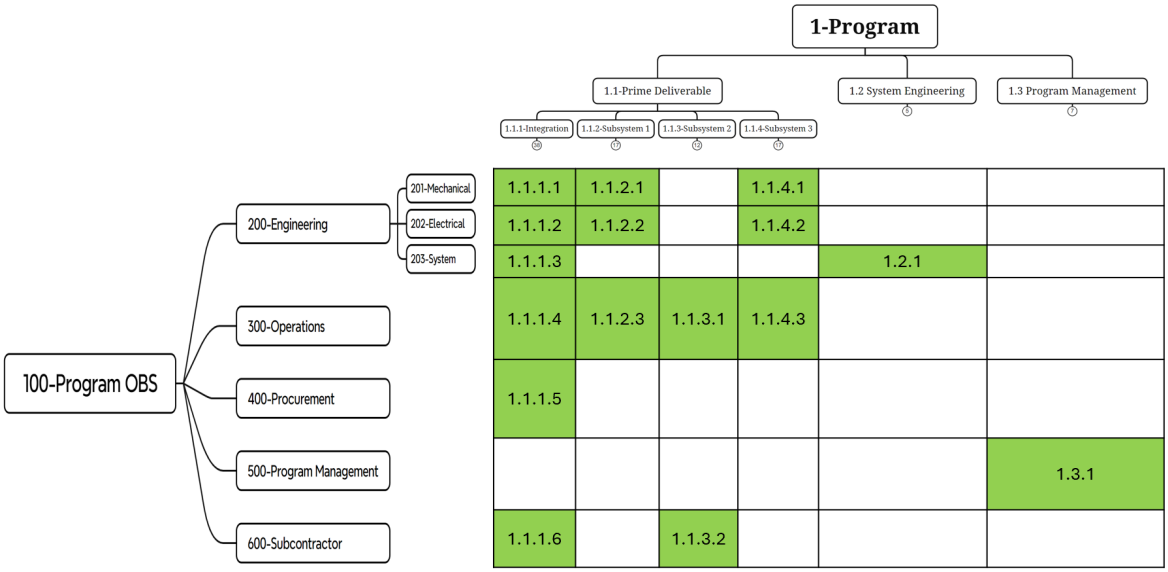


Figure 2-Example Program Initial Control Account Identification

Now that we've identified intersections, we can redraw the image and introduce our control accounts into our hybrid WBS as a unified structure. We will only expand the Integration branch of the WBS to simplify the diagram.

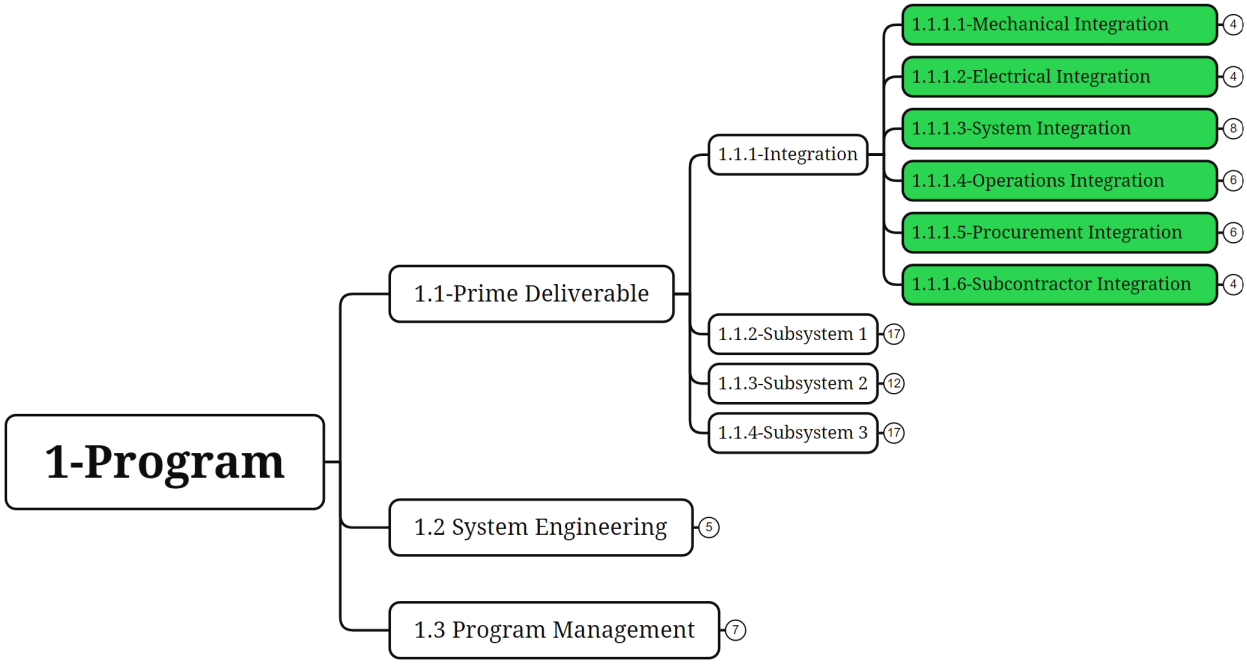


Figure 3-Integrated Control Accounts in Hybrid WBS

Now, we can introduce our first integration challenge. The contract is separated into four different Contract Line Items (CLINs). Let's see what happens when we ensure that CLIN information is segregated in the structure.

Handling CLINs: A Longstanding Conflict in Guidance

Across the major EVMS and project control guidance documents, there is no consistent direction for how to integrate Contract Line Item Numbers (CLINs) into program structures - especially the Work Breakdown Structure (WBS). In fact, the guidance often contradicts itself:

- **MIL-STD-881F** is unequivocal: the WBS must remain product-oriented and must *not* include contract or funding elements such as CLINs, SLINs, or R/NR designations. These are considered project control artifacts - not part of the technical breakdown.
- **IP2M METRR**, in contrast, emphasizes that contractual and accounting structures *should be integrated* with the WBS to support full traceability - suggesting that visibility to CLINs at the WBS level is a requirement for effective oversight.

- **EIA-748D** and the **DoD EVMIG** offer a middle ground, recommending the use of **common coding structures** to link scope, schedule, and cost across different domains - but they stop short of defining *at what level* those coding linkages should occur, or how they should be implemented.
- The **NDIA Intent Guide**, in **Guideline 1 – Define Work Scope (WBS)**, states that the WBS contains all contract line items and end items, and all elements specified for external reporting – a direct contradiction to MIL-STD881F. This language implies that **CLINs must be represented in the WBS**, at least to the extent required for reporting.

As a result, practitioners are often left to choose between two unsatisfying extremes: violating structural purity by embedding CLINs directly into the WBS, or losing traceability by ignoring contractual relationships altogether. The example below demonstrates what happens when CLINs are layered into our product-oriented WBS structure.

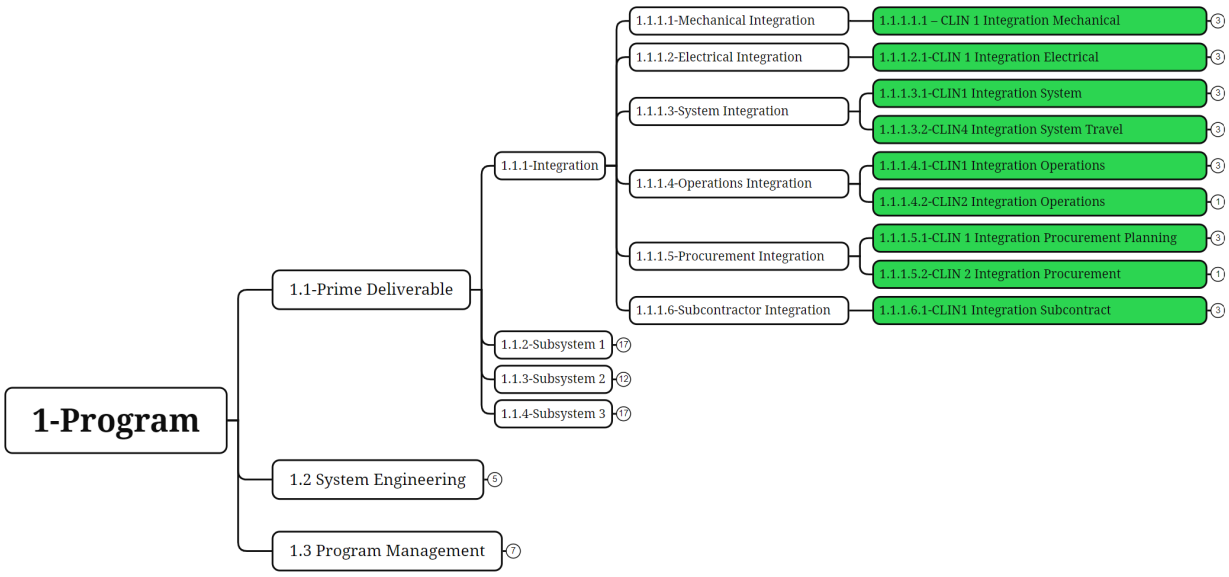


Figure 4-Integrated CLINs in Hybrid WBS

Our six integration control accounts expanded to nine to accommodate CLINs, and we've drilled down one more level into our hybrid WBS. Now let's see what happens when management adds a request.

Adding Phase Visibility: Complexity Compounds

To support internal management visibility, program leadership now requests that scope and budget be broken out by lifecycle phase - specifically: **Design, Test, Production**

Readiness, and Production. This is a common and valid request, especially for high-complexity programs with staged deliverables or milestone-driven reviews. However, because the WBS is already carrying CLINs as a structural layer, there's no clean place to add this new dimension.

The result: we add another level to the WBS to represent each phase under each CLIN node. In just one branch of the structure - originally supported by **6 control accounts** - we now have **23 control accounts**, and are continuing to add more. Each new dimension we try to represent structurally **multiplies the number of control accounts** exponentially.

This doesn't just increase setup complexity. It introduces **process overhead** in baseline change management, variance analysis, and reporting. Even worse, we've now broken the original purpose of the control account: it's no longer aligned with a single owner's scope and responsibility. Instead, it's serving as a reporting aggregation for contract, phase, and cost visibility - functions it was never designed to carry.

And we're still not done. What happens if a new CLIN is added midstream? Or if a fifth phase is introduced? Or what if leadership decides that recurring/non-recurring costs need to be tracked next? The structure is now **rigid, overloaded, and difficult to adapt.** As program leadership requests visibility by phase, we compound the problem. Each WBS element must now be repeated for each phase, multiplying control accounts again.

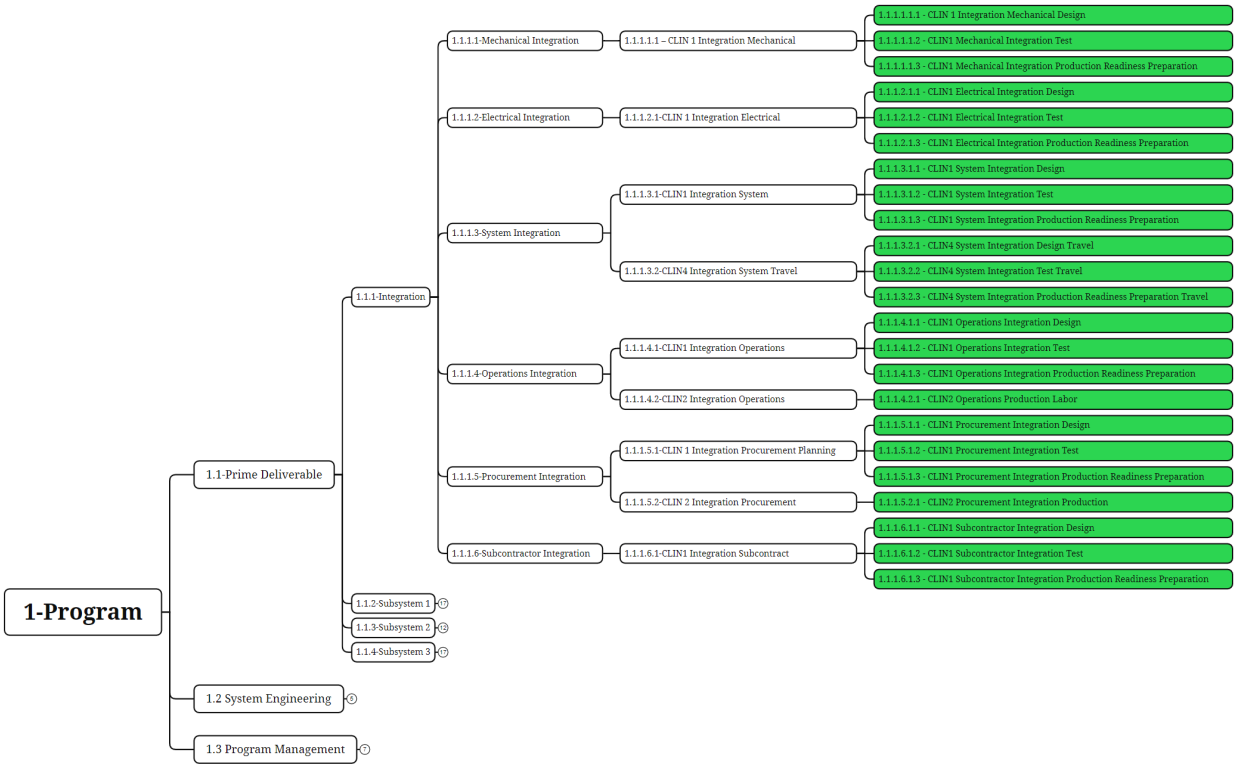


Figure 5-Hybrid WBS with CLIN and Phase breakout levels

From just six original control accounts, the structure expanded nearly four-fold, driven not by scope but by forced visibility into phases and CLINs. None of these splits support better control - they only exist to satisfy visibility needs that could be managed elsewhere.

When Structure Breaks Down

Even in this simplified example, we've introduced significant control and visibility problems that will only compound over time:

- **Control accounts are no longer clear management control points**. By forcing visibility structures (like CLINs and Phases) into the WBS, we've obscured the original WBS/OBS intersections that should define accountability. The result is ambiguity - who owns what is no longer obvious.
- **The WBS had to be extended unnaturally** to carry integration data. The product structure, once clean and compliant with MIL-STD-881F, is now burdened with contract and management reporting artifacts.
- **Control accounts have ballooned** from 6 to 23 in just one WBS branch - a nearly 400% increase. We've not yet accounted for cost elements, recurring/non-recurring splits, or other management concerns. Any one of those will inflate this further.

- **We still don’t know where cost collection should happen**. At what level will charge numbers align? How will we trace costs to CLINs or phases? The structure doesn’t tell us, and the control accounts can’t absorb it all.

This isn’t a minor reporting nuisance. The **control account** is the cornerstone of project control operations:

- It’s where **baseline change control** is exercised.
- It’s where formal **variance analysis and corrective actions are documented**.
- It’s the node for **work authorization and CAM responsibility**.

By overloading control accounts to solve integration problems, we’ve broken their intended function. We've lost the **management control** that control accounts are supposed to deliver.

The Core Problem: The WBS Is Two-Dimensional - The Data Is Not

Here lies the structural flaw at the center of every hybrid WBS attempt:

- The WBS is, by definition, a **two-dimensional product-oriented decomposition** of scope.
- But the information we care about - contractual structure, financial alignment, functional traceability, phase-based performance - is **multi-dimensional**.
- Yet major guidance documents often instruct us to “integrate to the WBS” without recognizing that this causes a collapse in scalability and clarity.

This contradiction - **"integrate everything into the WBS," but "don’t overload the WBS"** - puts programs in a no-win situation. The more integration you need, the more likely your structure will become rigid, complex, and unstable.

This breakdown is not caused by a lack of effort or intent - it’s caused by inconsistent structural definitions and a lack of clarity about where integration should occur. To move forward, we must first reestablish the foundational elements that underpin all program structures. That means clearly defining the roles and relationships of the WBS, OBS, control accounts, and work packages - before rebuilding them in a more resilient way.

Clarifying Common Structures

Before we can intelligently integrate across dimensions, we need to establish a clear, compliant foundation. That foundation is what IRDM calls the **Common Structure** - the set

of core organizing elements that appear consistently across all major EVMS and project management guidance documents. These elements are the building blocks of any structured project control system and form the root layer from which all other data relationships should be defined.

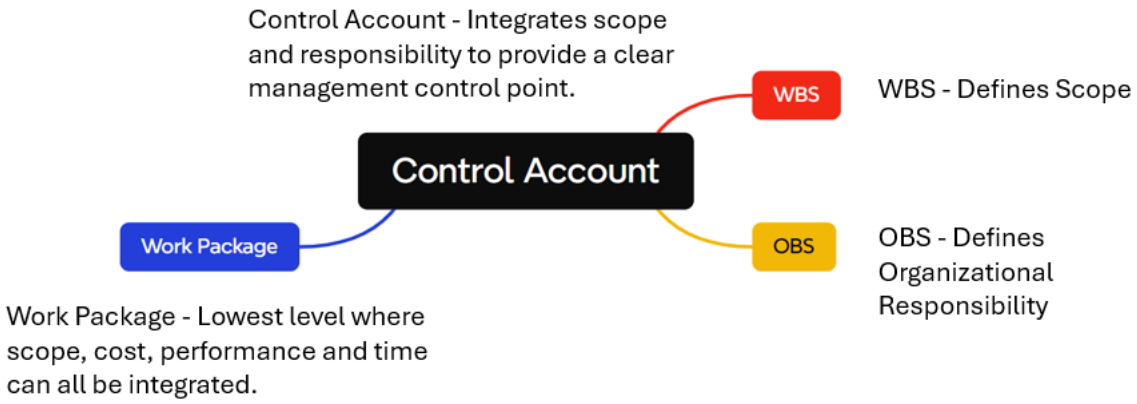


Figure 6-IRDM Common Structure

These four structures are present - explicitly or implicitly - in every formal guidance document, including EIA-748D, the NDIA Intent Guide, PASEG, and the DoD EVMIG.

Any effective integration strategy must start here.

The Role of Control Accounts and Work Packages

At the heart of every project control structure lies the relationship between the **control account** and the **work package**. These elements define the boundary between strategic control and tactical execution. To build a scalable, integrated system, we must first get this relationship right.

A **control account** is where responsibility meets scope. It is not a reporting bucket. It is not a placeholder for slicing visibility. It is a deliberate **point of management control** - the place where a Control Account Manager (CAM) is accountable for performance, baseline integrity, and corrective action.

A good structure:

- Places control accounts only where **actual authority and responsibility exist**
- **Avoids inflating control accounts** to satisfy contract or reporting visibility
- Keeps the number of control accounts **optimized** to minimize process overhead (e.g., BCRs, variance analysis, work authorization)

The **work package** is the operational building block of IRDM. It is the lowest-level planning and control element - where scope, budget, schedule, and earned value performance are defined and measured.

Each work package:

- Belongs to **exactly one control account**
- Is the level at which **cost, schedule, performance, and forecasting** converge
- Carries the **metadata** that links it to all relevant integration points: CLINs, cost elements, recurring/non-recurring designation, phase, earned value method, functional area, and more

Instead of embedding complexity into the control account or WBS, this approach distributes it across clearly structured work packages - each traceable, filterable, and tailored to meet program integration and reporting needs.

Work Packages vs. Tasks

Before moving forward, it's important to address a common source of confusion in the guidance documents: the inconsistent use of terms like "work package," "activity," and "task." While these are sometimes used interchangeably, they serve fundamentally different roles in a well-structured control system.

- A **work package** is the lowest level of cost and performance measurement in the project controls hierarchy. It represents a **block of scope, budget, schedule, and earned value** and is the point at which planning and control converge.
- A **task** (or schedule activity), by contrast, is a time-based unit of execution. It appears in the Integrated Master Schedule (IMS) and is used to track progress, dependencies, and float. Tasks exist to **fulfill** the scope of a work package.

A clear distinction must be made: control accounts manage work packages, and work packages may contain one or more tasks. While tasks can be linked to specific artifacts like the Integrated Master Plan (IMP), the work package is the essential integration point where cost, schedule, and performance converge. This is the level at which key metadata—such as CLINs, cost elements, EV methods, and functional ownership—is most effectively applied. The table below summarizes the distinct roles and attributes of work packages

and tasks, clarifying how each supports different aspects of program control, scheduling, and performance management.

Attribute	Work Package	Task
Definition	Lowest-level planning element for performance, cost, and schedule integration.	Scheduling activity or group of activities within a work package.
Ownership	Belongs to a single control account.	Belongs to a single work package.
Scope Type	Represents a definable segment of scope with assigned budget and schedule.	Represents effort or activity to execute part of the work package.
Schedule Linkage	May or may not be directly scheduled.	Is always scheduled.
Earned Value Measurement	Has an assigned EV method and measurable performance baseline.	Supports schedule logic and milestone tracking.
Metadata Integration	Carries metadata (e.g., CLIN, cost element, phase, EV method, R/NR, etc.).	Does not independently carry EVMS metadata.
Change Control Impact	Changes require formal control account-level authorization.	Typically managed within the scheduling system.
Primary Purpose	Organizing unit for EVM scope, budget, schedule, and performance reporting.	Logical building block for detailed schedule sequencing and analysis.
Reporting Granularity	Basis for EVM reporting, variance analysis, and forecasting.	Basis for detailed schedule review and network analysis.

Table 1-Compare the Attributes of Work Packages and Tasks

Grounding the Structure in Clear Definitions

Establishing precise definitions for common structure elements isn't just academic - definition gaps directly lead to overbuilt control account hierarchies, blurred accountability, inconsistent reporting logic, and avoidable audit findings. Before we can build an integrated system or propose a framework for solving the integration problem, we must **re-anchor the meaning and function** of each common structure element. With these definitions established, we're ready to establish a modern and effective project control structure.

With a shared understanding of the core structural elements and the dimensions that drive complexity, we can now turn to the method that ties them together. The Integration Relationship Diagramming Method (IRDM) provides a repeatable, four-step process for building integrated program structures that are scalable, compliant, and traceable by design.

Introducing the Integration Relationship Diagramming Method

The Integration Relationship Diagramming Method (IRDM) is a practical framework for building scalable, standards-aligned project control systems. It resolves long-standing contradictions in integrated program management guidance by shifting integration from hierarchy to metadata, and by establishing explicit relationships between structural elements rather than relying on inference or overbuilt WBS models.

To streamline implementation and support repeatability across diverse program environments, the IRDM process is organized into four core steps. These steps represent the logical and structural flow needed to establish clarity, traceability, and control in any program:

1. Define and Relate Structural Elements

This step establishes the four foundational structures that exist in every program, whether formalized or not:

- **Common Structure**: WBS, OBS, Control Accounts, Work Packages
- **Contract Structure**: CLINs, SLINs, ACRNs, Delivery Orders
- **Accounting Structure**: Cost collection hierarchies, charge number schemas, business rules
- **Program and Business-Specific Structure**: Phases, cost elements, EV methods, recurring/non-recurring splits, internal reporting dimensions

Each structure is diagrammed independently to maintain clarity. Relationships between elements - such as whether work packages or control accounts are the element aligned with CLINs, or where the accounting system maps to work scope - are defined explicitly, not embedded through hierarchy.

With all key program dimensions identified and assigned to one of the four organizing structures, the next task is to translate those structures into an actionable common framework. That begins with building a clean, compliant Common Structure.

2. Determine Common Structure

This step defines the core execution layer of the program: the WBS, OBS, Control Accounts, and Work Packages. Here, we:

- Build a product-oriented WBS
- Create an OBS that reflects true responsibility and accountability

- Define control accounts at the intersection of WBS and OBS where actual management control is exercised
- Generate work packages that carry integration metadata and form the point of measurement

This layer becomes the backbone for all future reporting and system behavior. With our WBS, OBS, control accounts, and work packages defined, we now need to make the structure operational. This is where naming conventions come into play - adding traceability, automation, and human readability to every element.

3. Apply Coding and Naming Conventions

Structured, metadata-driven naming conventions make the system navigable, filterable, and automatable. In this step, we:

- Apply a consistent and intelligent coding scheme across WBS, OBS, control accounts, and work packages
- Embed integration metadata directly into work package codes (e.g., CLIN, phase, cost element)
- Enable native sortability and traceability in tools like Excel and Power BI

This step eliminates reliance on lookup tables and ensures that every code is meaningful at a glance - whether read by a human or parsed by a system. With codes applied and metadata integrated, the final step is validation. This ensures our control account definitions support clear ownership, align with earned value methods, and meet all internal and external reporting obligations.

4. Validate and Review Program Structure

In the final step, we ensure the structure is not only technically sound but operationally usable and compliant. We:

- Confirm each control account has a single, clearly defined CAM
- Split or consolidate control accounts if ownership, scope, or EV method requirements demand it
- Validate that all programmatic, contractual, and financial reporting needs can be met through metadata and filtering - not hierarchy manipulation

The goal is to ensure that the structure supports execution, not just satisfies documentation.

Together, these four steps provide a disciplined, scalable alternative to legacy organizing models. IRDM doesn't abandon the standards - it implements their intent while correcting for structural conflicts that have gone unresolved for decades.

Applying IRDM to our Example Program

With this framework in place, we can now reapply it - step by step - to the same example that broke down under a hybrid WBS. This walkthrough not only illustrates the four steps in action but shows how intentional integration prevents structural failure and supports scalable reporting.

We'll use the exact same program elements - scope, WBS, OBS, CLINs, phases, and cost visibility requirements - but apply a cleaner, more deliberate strategy. Rather than forcing these dimensions into the WBS or Control Account hierarchy, we'll preserve the integrity of the **Common Structure**, define clear relationships across domains, and manage complexity through metadata - not hierarchy.

IRDM is applied in four logical steps. In the sections that follow, we'll walk through each of these steps in context, using our example program to demonstrate how IRDM scales integration without sacrificing control or compliance.

But before we begin, we need to establish a shared understanding of the **structural elements that IRDM connects**. That means clarifying the four types of organizing structures that must be defined for any program - regardless of size, contract type, or reporting environment.

Step 1 - Define and Relate Structural Elements

The Four Organizing Structures of IRDM

To fully support complex program operations, IRDM includes a framework for identifying and classifying **all of the dimensions** that influence scope, cost, responsibility, and control.

These dimensions are not arbitrary - they fall into one of **four distinct organizing structures**, each playing a specific role in shaping how the program is understood, managed, and reported.

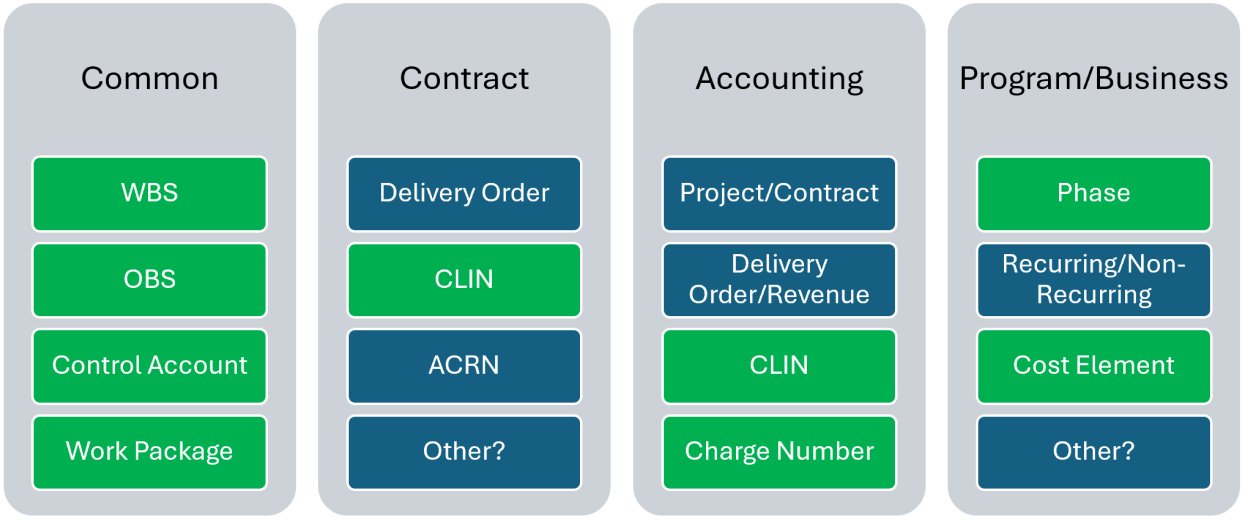


Figure 7-The Four Structures that Generate Integration Requirements – Green highlights elements used in the example program, while blue shows potential elements that may be used on other programs

1. Common Structure

This is the backbone of every program:

- **WBS** – defines the work
- **OBS** – defines responsibility
- **Control Accounts** – where scope and responsibility meet
- **Work Packages** – the atomic unit for planning, execution, and performance measurement

These elements are **always present** in IRDM and form the structure upon which all integration is layered.

2. Contract Structure

These are the integration points **required by the contract** - whether for reporting, invoicing, or funding allocation. Common examples include:

- **CLINs (Contract Line Items)**
- **Delivery or Task Orders**
- **SLINs (Sub Line Items)**
- **ACRNs (Accounting Classification Reference Numbers)**

These define what the customer contracting officer expects to track.

3. Accounting Structure

Every business has its own internal cost collection system, often multi-layered. The accounting structure maps how a program aligns with that system. Examples include:

- Charge number schemas
- Revenue recognition codes
- Business unit or profit center hierarchies
  This structure is **organization-specific** but must be reflected in the integration plan.

4. Program and Business-Specific Structure

These are items driven by **program-specific scope**, internal visibility requirements, or even customer preferences. Examples:

- Phases (Design, Test, Production)
- Recurring vs. Non-Recurring cost designations
- Cost Element classifications (Labor, Travel, Subcontract)
- EV Method
- Departmental Information
- Historical Basis of Estimate categorization

These dimensions vary by program but are essential for effective management and performance analysis.

Diagramming Common Structure: Establishing Parentage and Hierarchy

Now that we've identified the four integration domains, it's time to transition into the **diagramming portion** of the Integration Relationship Diagramming Method.

We begin with the **Common Structure** - the core elements every program shares: WBS, OBS, Control Accounts, and Work Packages.

In IRDM, we use a **solid arrow** to represent a clear parent-child relationship within a structure. This visual convention helps communicate which elements **own or govern** others - and which ones remain **parallel but connected**.

In the diagram below, note the following relationships:

- **WBS → Control Account**: Each control account belongs to a single WBS element.

- **OBS → Control Account**: Each control account is also aligned to a single OBS element (representing responsibility).
- **Control Account → Work Package**: Every work package has one - and only one - control account parent.

Importantly, **WBS and OBS are not linked to each other directly** - they are independent hierarchies that converge at the control account. This distinction preserves flexibility and avoids conflating scope and responsibility inappropriately.

This visual foundation will allow us to now build and link other structures - contract, accounting, and program-specific - using similar conventions.

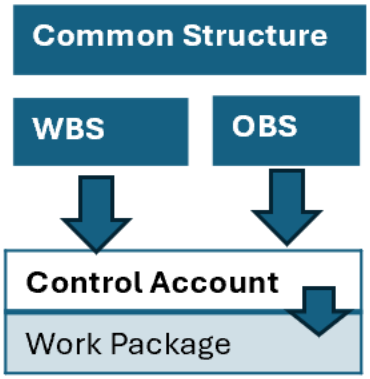


Figure 8-Common Structure Diagram

Diagramming Contract Structure: Mapping Customer-Driven Requirements

The second IRDM structure we diagram is the **Contract Structure** - the organizing system required to satisfy contractual obligations around reporting, funding, and invoicing. These elements do not exist to manage internal execution - they exist because **the customer requires them**.

Just like in the Common Structure, we use **solid arrows to show parent-child relationships**. But here, the hierarchies are purely contractual:

- **Delivery Orders** may contain multiple CLINs (Contract Line Item Numbers)
- **CLINs** may contain **Sub-CLINs**, or break down into **ACRNs** (Accounting Classification Reference Numbers) and **AAIs** (Authorized Accounting Instructions)

Not every program uses all of these layers. The structure is **flexible** and reflects what the contract stipulates.

What matters in IRDM is this:

- These are **not embedded into the WBS**
- These are **not embedded into the OBS**
- They are **independent structures** that need to be linked intentionally to the Common Structure through explicit relationships

By diagramming the contract structure separately, we preserve **traceability without compromising the core architecture** of the program's work and responsibility models.

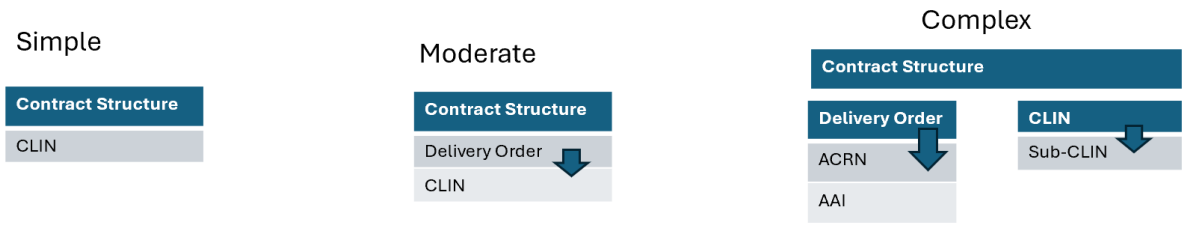


Figure 9-Contract Structure Diagram

Diagramming Accounting Structure: Aligning With Financial Systems

The third structural domain in IRDM is the **Accounting Structure**. This represents how the organization tracks, collects, and reports financial data through its enterprise systems. Unlike the Common and Contract structures, which are governed by project execution and contractual compliance, respectively, the accounting structure is **unique to each business** - and can even differ by business unit or division within the same organization.

This structure governs:

- **Billing**
- **Revenue recognition**
- **Financial EACs**
- **Charge number configuration**
- **Audit trail for cost control and actuals tracking**

The diagram below illustrates a **four-level hierarchy** commonly seen in federal contractors:

- **Project/Contract**: The top-level financial tracking point

- **Delivery Order/Revenue Recognition Unit**: Aligned to payment and revenue triggers
- **CLIN**: Often used for internal budgetary tracking
- **Charging/Cost Collection Level**: The endpoint that maps to charge numbers where cost is collected and timekeeping established

In IRDM, we again use **solid arrows** to indicate parent-child relationships. But we make no assumption about how this structure maps to the WBS or Control Accounts - **that mapping must be explicitly defined** during integration.

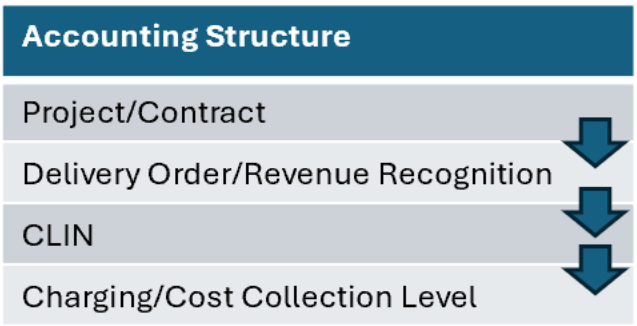


Figure 10-Accounting Structure Diagram

Diagramming Program and Business-Specific Structure: Capturing Operational Nuance

The final dimension in the IRDM framework is the **Program and Business Specific Structure**. These elements may not be required by contract or dictated by the financial system - but they are critical to how the **business or program wants to view and manage the work**. This structure also supports customer-specific visibility requests that are not explicitly defined by contract. **This is a very important capability, as it allows a program team and customer to agree on reporting visibility without introducing contractual complexity in the CLIN structure**.

These structures originate from:

- Internal leadership visibility needs
- Program Management concerns
- Historical Basis of Estimate data capture and organization
- Statement of Work deliverables
- Data reporting obligations like **CSDR** (Cost and Software Data Reporting)

- Customer requested visibility items

This structure is especially important for **recurring vs. non-recurring splits**, **functional reporting**, and **management insight into specific program attributes** like phase, site, configuration, or subsystem. **This structure gives the program the flexibility to define and ensure data visibility that directly answers critical management questions.**

As shown in the diagram below, this structure can be simple - tracking a single dimension like phase - or incredibly complex, cascading across multiple attributes like vehicle identifier, build site, or system variant.

In IRDM, this structure is treated **on equal footing** with the others and mapped explicitly - not buried in the WBS or approximated through post hoc analysis.

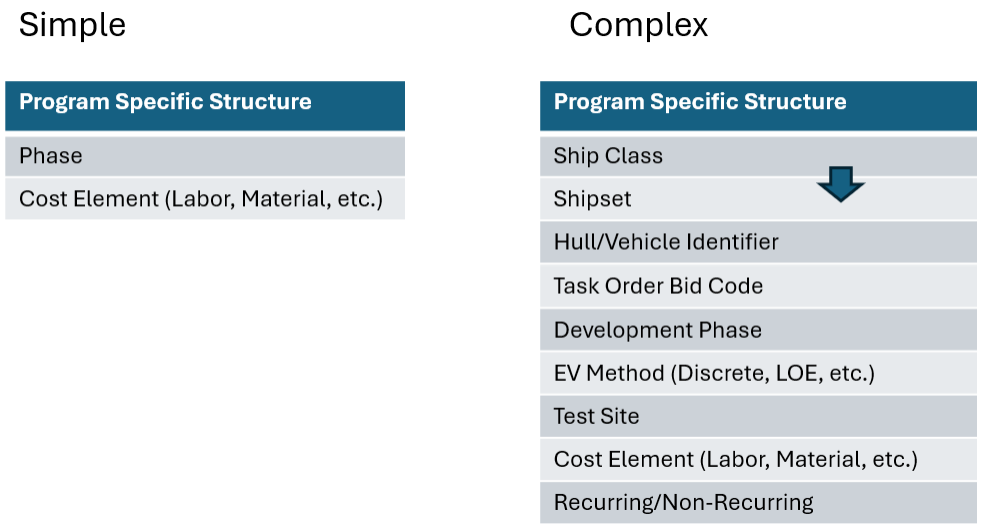


Figure 11-Simple and Complex Example Program Structure Diagrams

Introducing the Integration Relationship Diagram (IRD)

Now that we've defined each of the four organizing structures - Common, Contract, Accounting, and Program-Specific - we're ready to bring them together visually. The image below shows the starting point of the **Integration Relationship Diagram** for our example program.

Each structure is built independently based on its own logic and governance. The **Common Structure** defines the WBS, OBS, Control Accounts, and Work Packages. The **Contract Structure** captures funding elements like CLINs. The **Accounting Structure** supports cost collection and financial reporting. And the **Program-Specific Structure** enables unique operational or reporting views, such as development phase.

At this stage, no connections between the structures have been made. Each one is complete on its own terms - but not yet integrated. What comes next is where the IRDM method proves its value: we'll define **explicit relationships between these structures**, identifying where they intersect and how they will be tied together through metadata and naming conventions.

Each organizing structure - Common, Contract, Accounting, and Program-Specific - exists independently, aligned to its own logic and ownership.

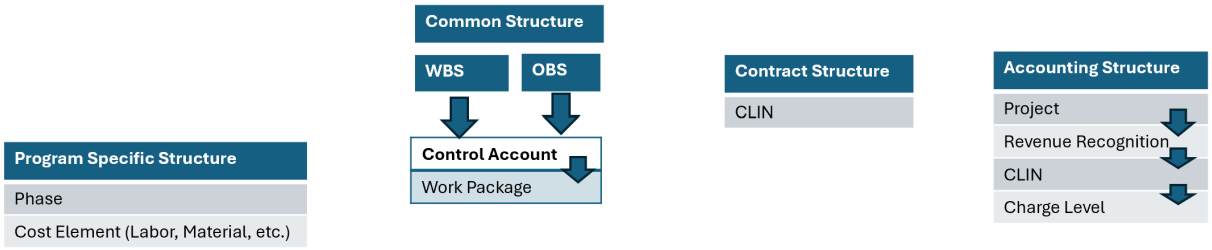


Figure 12-The Integration Relationship Diagram Before Establishing Relationships

At this stage, complexity is defined but not yet integrated. The structure is stable, but unconnected.

Defining the Right Integration Point for CLINs

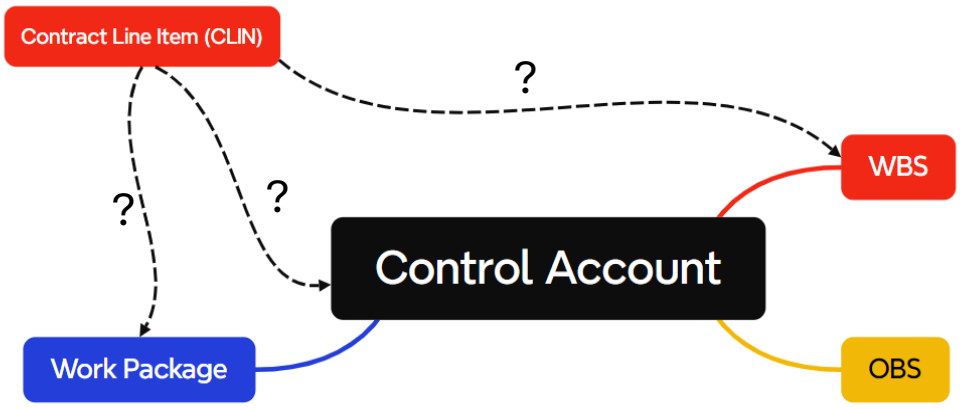


Figure 13-Determining the Level of Integration

With our WBS and OBS intersections established, the next step is integrating contract-level structure - specifically, CLINs. Under IRDM, we explicitly define where this relationship lives. As shown in the diagram, there are three potential integration points: the WBS, the

Control Account, or the Work Package. We've already seen the structural damage caused by embedding CLINs into the WBS. Integrating at the Control Account level may be viable - but only if CLINs align cleanly with WBS/OBS intersections, which is rare. The most flexible and scalable choice is to integrate CLINs at the **Work Package** level. This allows us to add CLIN visibility **without impacting structure or process overhead**.

Completing the Integration Relationship Diagram

Now that we've built out each structure and defined the data elements within them, we can complete step one of IRDM by drawing **explicit integration links**. The diagram below illustrates the complete **Integration Relationship Diagram** for our example program, showing how the structures interconnect through intentional integration decisions.

These arrows represent directional relationships and carry an important logic: **a data element cannot be split across more than one of its integration targets**. That is, if we map a Work Package to a Phase, it cannot straddle two different Phases. If a Charge Number integrates to a Work Package, it can belong to only one. These integration rules are the backbone of a reliable and scalable program control structure.

In our example:

- A **Work Package** cannot be split across more than one **Phase**.
- A **Charge Number** (from the Accounting Structure) cannot span multiple **Work Packages**.
- A **CLIN** (from the Contract Structure) must have a one-to-one mapping at its integration level.

This model avoids ambiguity, enforces clean joins across disparate data sources, and allows metadata to flow predictably into reports and analysis - all while minimizing the complexity of your control account structure. Now we have a ruleset to follow for any element we define in the program. This is the power of IRDM: deliberate connections, not accidental entanglements.

With relationships defined, we intentionally connect each structure to its integration targets using metadata and work packages - not hierarchy.

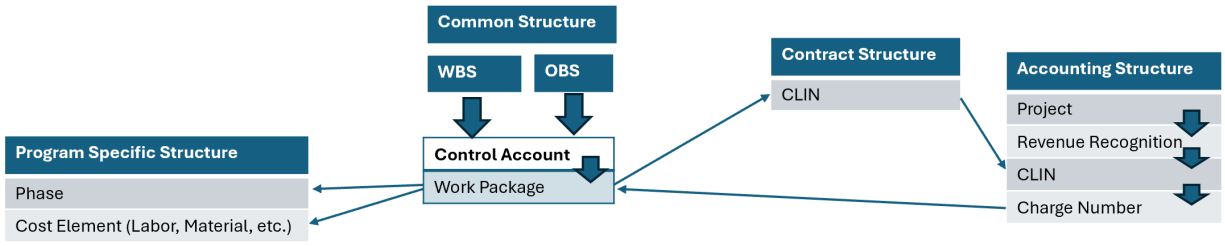


Figure 14-The Final Integration Relationship Diagram for our Example Program

Integration is no longer embedded. It's explicit, flexible, and scalable - mapped once, reused everywhere.

Step 2 – Determine Common Structure

We begin exactly where we started before - with a clean definition of control accounts at the intersection of the WBS and OBS. This time, we'll maintain that structure throughout the exercise. Notice how the first step is exactly the same as our Hybrid WBS example.

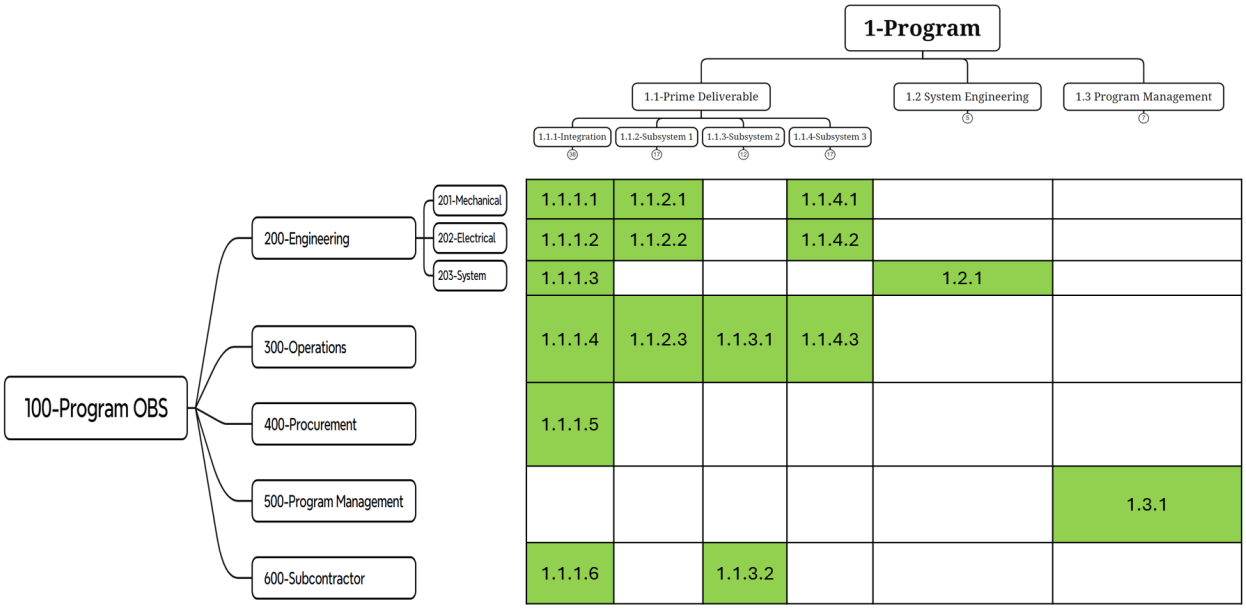


Figure 15-Assigning Control Accounts with IRDM

This is our foundation. Everything that follows - contract structure, cost structure, program structure - will build on this without corrupting it.

Integrating CLINs Without Expanding Control Accounts

Using IRDM principles, we now introduce CLIN visibility into the program while holding the number of control accounts steady at six. Rather than assigning a separate control account

for each CLIN, we define a set of **work packages beneath each control account**, with each work package uniquely tagged to its respective CLIN.

While this initial representation shows CLINs appearing at a deeper WBS code level, we are not adding new WBS elements to the product-oriented hierarchy - we're simply extending the **WBS naming scheme** to distinguish work packages for clarity. The structural integrity of the WBS remains intact, and all integration occurs at the work package level through metadata.

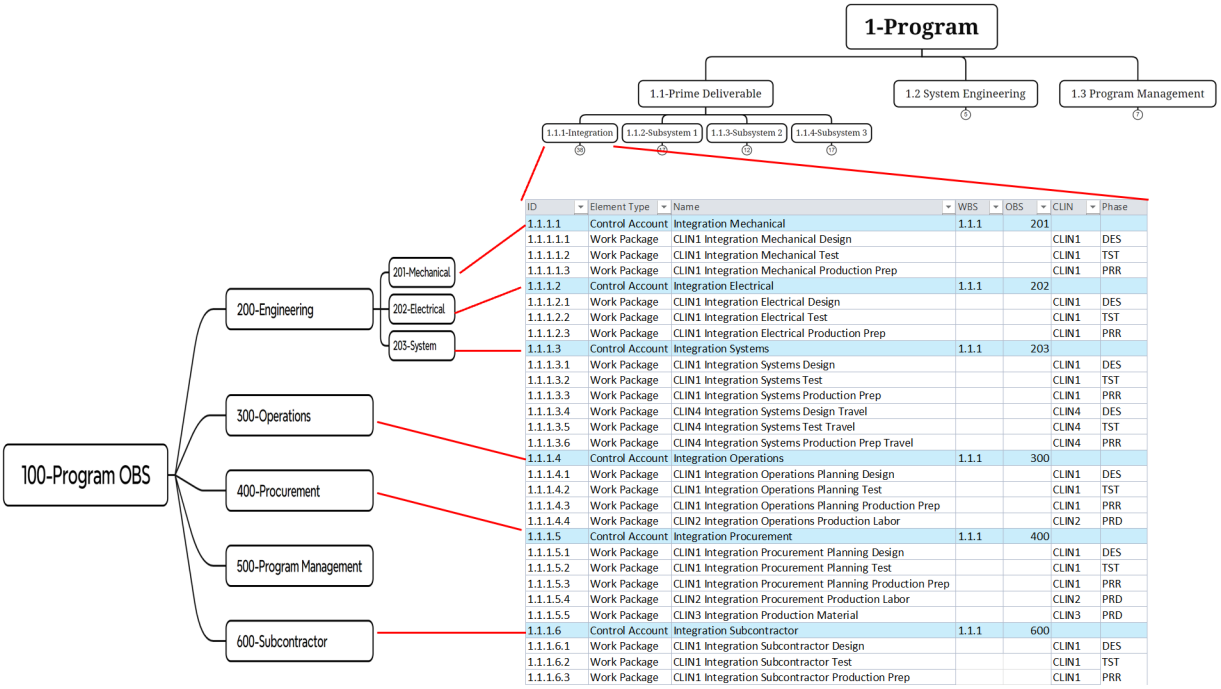
The result: we achieve full CLIN traceability across scope, schedule, and cost with **zero increase in control accounts**.

Reintroducing Phase Visibility - Without Structural Penalty

With CLINs already integrated cleanly through work packages, we now reintroduce phase visibility - Design, Test, Production Readiness, and Production. Under the IRDM approach, this does **not** require structural expansion or additional control accounts. Instead, each required phase is represented by a dedicated work package under the appropriate control account, using **metadata to define phase alignment**.

In this example, the structure still contains **only 6 control accounts**, but now supports **24 work packages**, each mapped to a specific CLIN and phase. This enables detailed performance tracking and reporting across both contractual and programmatic dimensions - without compromising the original WBS/OBS alignment or inflating the control account structure.

What was previously a rigid, overloaded hierarchy is now a **flexible, modular design**. Integration decisions are deliberate. Visibility is preserved. And process overhead remains stable.



ID	Element Type	Name	WBS	OBS	CLIN	Phase
1.1.1.1	Control Account	Integration Mechanical	1.1.1	201		
1.1.1.1.1	Work Package	CLIN1 Integration Mechanical Design			CLIN1	DES
1.1.1.1.2	Work Package	CLIN1 Integration Mechanical Test			CLIN1	TST
1.1.1.1.3	Work Package	CLIN1 Integration Mechanical Production Prep			CLIN1	PRR
1.1.1.2	Control Account	Integration Electrical	1.1.1	202		
1.1.1.2.1	Work Package	CLIN1 Integration Electrical Design			CLIN1	DES
1.1.1.2.2	Work Package	CLIN1 Integration Electrical Test			CLIN1	TST
1.1.1.2.3	Work Package	CLIN1 Integration Electrical Production Prep			CLIN1	PRR
1.1.1.3	Control Account	Integration Systems	1.1.1	203		
1.1.1.3.1	Work Package	CLIN1 Integration Systems Design			CLIN1	DES
1.1.1.3.2	Work Package	CLIN1 Integration Systems Test			CLIN1	TST
1.1.1.3.3	Work Package	CLIN1 Integration Systems Production Prep			CLIN1	PRR
1.1.1.3.4	Work Package	CLIN4 Integration Systems Design Travel			CLIN4	DES
1.1.1.3.5	Work Package	CLIN4 Integration Systems Test Travel			CLIN4	TST
1.1.1.3.6	Work Package	CLIN4 Integration Systems Production Prep Travel			CLIN4	PRR
1.1.1.4	Control Account	Integration Operations	1.1.1	300		
1.1.1.4.1	Work Package	CLIN1 Integration Operations Planning Design			CLIN1	DES
1.1.1.4.2	Work Package	CLIN1 Integration Operations Planning Test			CLIN1	TST
1.1.1.4.3	Work Package	CLIN1 Integration Operations Planning Production Prep			CLIN1	PRR
1.1.1.4.4	Work Package	CLIN2 Integration Operations Production Labor			CLIN2	PRD
1.1.1.5	Control Account	Integration Procurement	1.1.1	400		
1.1.1.5.1	Work Package	CLIN1 Integration Procurement Planning Design			CLIN1	DES
1.1.1.5.2	Work Package	CLIN1 Integration Procurement Planning Test			CLIN1	TST
1.1.1.5.3	Work Package	CLIN1 Integration Procurement Planning Production Prep			CLIN1	PRR
1.1.1.5.4	Work Package	CLIN2 Integration Procurement Production Labor			CLIN2	PRD
1.1.1.5.5	Work Package	CLIN3 Integration Production Material			CLIN3	PRD
1.1.1.6	Control Account	Integration Subcontractor	1.1.1	600		
1.1.1.6.1	Work Package	CLIN1 Integration Subcontractor Design			CLIN1	DES
1.1.1.6.2	Work Package	CLIN1 Integration Subcontractor Test			CLIN1	TST
1.1.1.6.3	Work Package	CLIN1 Integration Subcontractor Production Prep			CLIN1	PRR

Figure 16-IRDM Adding Work Packages While Holding Control Accounts Stable

So far, we've completely rebuilt our integration branch using IRDM principles - and the results speak for themselves:

- The **WBS stops at level 3**, maintaining structural purity and alignment with MIL-STD-881F.
- Control accounts are held at the **right level** for management responsibility - no inflation, no ambiguity.
- We've avoided a **400% increase in control account count**, sparing the program from significant baseline change, work authorization, and variance analysis overhead.
- We've integrated CLINs, phases, and programmatic structure by adding targeted **work packages**, each tagged with the metadata needed for reporting and decision support.

That's a strong foundation - but we can go further.

Step 3 - Apply Coding and Naming Conventions

Right now, we can't easily distinguish between element types by looking at their identifiers. Are we filtering a control account or a work package? Are we looking at a rollup code or an

execution-level code? Even in structured tools like Excel or Power BI, this lack of **human readability** adds friction to analysis and increases the chance of error.

The next step in our journey is about **clarity and usability**. By applying structured, rule-based **naming conventions**, we bring meaning to every code. We make element types distinguishable at a glance, enable intuitive sorting and filtering, and make program data far more self-explanatory and maintainable.

Naming Rules: Adding Context, Clarity, and Control

With integration handled cleanly at the work package level, another enhancement in IRDM is to make the structure more readable, maintainable, and self-explanatory through consistent naming conventions.

Naming isn't just cosmetic. It's a **mechanism of control**. Properly structured codes allow you to tell what something is, what it belongs to, and what kind of data it holds - **at a glance**. When applied consistently, naming conventions reduce ambiguity, enhance analysis, and make the entire system easier to navigate for everyone from CAMs to executives to auditors. To make structure self-explanatory and system-friendly, IRDM applies a set of consistent, fixed-position naming rules for all structural elements.

Below is a summary of the **six core naming rules** used in IRDM, along with the **reason each rule matters**:

IRDM Naming Rules and Why They Matter

Rule	Description	Why It Matters
Element types have different schemas	WBS = 1.1.2, OBS = PGM.PMO, Control Account = 111PMO1, Work Package = 111PMO1.L1N1	We often view data out of hierarchy; we need to know if a code is WBS, OBS, CA, or WP without guessing.
Use numbers for WBS, letters for OBS	Keeps structure types visually distinct. OBS elements must have unique bottom-level codes.	Reinforces separation between product and responsibility structures - helps with auto-sorting and interpretation.
Child elements name their parents	Control accounts show WBS and OBS in the code. Work packages inherit CA code.	Enables fast rollup, intuitive tracing, and one-click filtering by parent in flat views.
Make metadata visible in work package codes	Use structured suffixes for CLIN, cost element, R/NR, phase, etc.	Allows metadata integration without extra columns - eases filtering and mapping to charge numbers.

Use fixed lengths for control account and WP codes	In our example program, Control account = 7 chars; work package suffix = 4 chars Actual code length configured by program.	Supports Excel formulas, parsing, and Power BI report automation. Each character has a defined role.
Plan for alphabetical order	Codes are structured to sort naturally by hierarchy	This enables faster reviews and native ordering in tools like Excel, SharePoint, and Power BI dashboards.

Table 2- IRDM naming conventions make structure self-documenting. Codes are designed to be readable by humans, sortable by machines, and consistent across all programs.

These rules don't just organize codes - they embed structural logic that supports sorting, filtering, and traceability across systems and tools.

Applying Naming Rules: Work Breakdown Structure (WBS)

We begin our element-by-element application of IRDM naming rules with the Work Breakdown Structure.

Per guidance from MIL-STD-881F and consistent with EIA-748D, WBS codes are constructed using **numbers separated by decimal points**, with each level denoting a new layer of hierarchical decomposition. This approach is universally recognized and makes scope rollup intuitive.

But not all numeric structures behave well in software environments. Once a WBS level reaches 10+ elements, simple numeric ordering can lead to sorting errors - especially in flat views like Excel or PowerBI. Consider the following scenarios:

- **Option 1**: Use **letters after 9** (e.g., 1.1.9, 1.1.A, 1.1.B)
  This improves **readability** and preserves **natural sorting** across reporting tools.
- **Option 2**: Use **leading zeroes** (e.g., 1.1.01 to 1.1.12)
  Also maintains sortability but can increase code length and reduce clarity in derivative codes.
- **What to avoid**: Unpadded numeric codes (e.g., 1.1.10, 1.1.11, 1.1.2)
  These **break alphabetical sort order** in tools like Excel and Power BI, creating data quality and filtering issues downstream.

IRDM Recommendation: Option 1 - use letters beyond 9 for WBS leaf codes. This approach keeps codes short, human-readable, and structurally consistent with derived control account and work package codes.

Option 1 – (Preferred)	Option 2 – (Acceptable)	Historical – Breaks alphabetic sorting rule
1	1	1
1.1	1.1	1.1
1.1.1	1.1.01	1.1.1
1.1.2	1.1.02	1.1.10
1.1.3	1.1.03	1.1.11
1.1.4	1.1.04	1.1.12
1.1.5	1.1.05	1.1.2
1.1.6	1.1.06	1.1.3
1.1.7	1.1.07	1.1.4
1.1.8	1.1.08	1.1.5
1.1.9	1.1.09	1.1.6
1.1.A	1.1.10	1.1.7
1.1.B	1.1.11	1.1.8
1.1.C	1.1.12	1.1.9

Figure 17-WBS Coding Options

Organizational Breakdown Structure (OBS): Clear Responsibility Coding

Next, we turn to the **Organizational Breakdown Structure (OBS)** - the dimension that defines "who" is responsible for delivering the work. Just like the WBS defines the work to be done, the OBS assigns ownership and accountability.

Under IRDM, we switch from numeric to **character-based coding** for the OBS. This immediately distinguishes OBS elements from WBS elements in tables, exports, and reports - eliminating confusion and enhancing readability.

Each OBS level is separated by a decimal, mirroring the WBS structure. We recommend using **three-character codes** for each level (e.g., PGM.ENG.MEC). The final, bottom-level OBS code - assigned to a Control Account Manager - **must be unique** across the entire OBS hierarchy. This uniqueness allows us to embed it directly into the **control account code**, preserving traceability and eliminating ambiguity.

By adopting a character-based, hierarchical scheme:

- Analysts can **instantly recognize** OBS elements by structure.
- **Responsibility roll-ups** become more intuitive.
- Control account naming becomes more structured and informative.

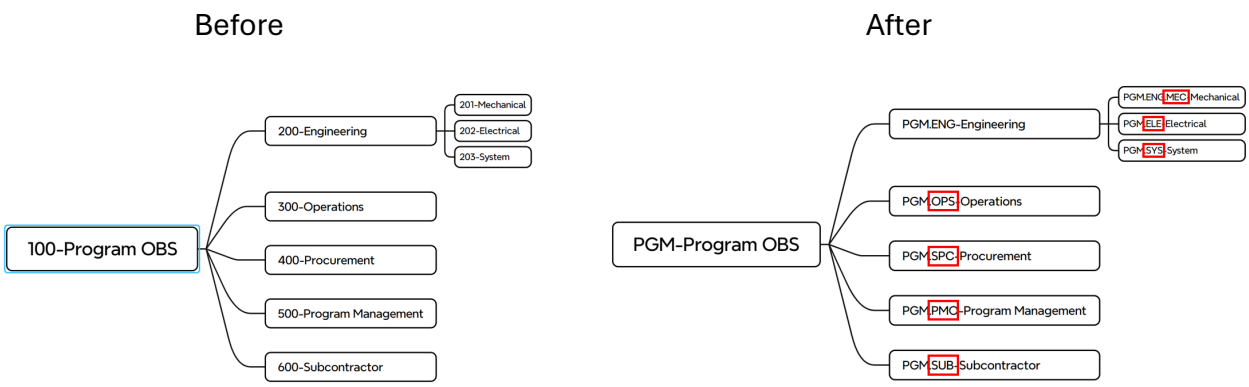


Figure 18-OBS Codes with IRDM Naming Conventions

Control Accounts: Structured, Scalable, and Sortable

Control accounts are the heart of earned value management - they represent the point where scope (WBS) and responsibility (OBS) intersect, and where performance is planned, measured, and managed.

In IRDM, we create control account codes by combining three elements:

1. **The WBS code**, with all decimal points removed.
2. **The bottom-level, three-character OBS code**.
3. **A numeric index**, used to distinguish multiple control accounts at the same WBS/OBS intersection.

This structure yields control account codes like:

WBS	OBS	#	Control Account Code
1.1.1	PGM.ENG.MEC	1	111MEC1
1.1.1	PGM.ENG.SYS	1	111SYS1
1.3	PGM.PMO	1	130PMO1

Table 3-IRDM Control Account Code Generation Example

Note the third example: 1.3 becomes 130 to maintain the **fixed-length rule**. This ensures that every character in the control account code occupies a consistent position, enabling formulas, filters, and parsing logic in Excel, Power BI, and SharePoint. It also allows shorter branches (e.g., Program Management) to coexist cleanly with deeper ones (e.g., detailed engineering products). A flat list of control accounts will now natively sort in WBS order.

For example:

- 112131ENG1: Engineering for subsystem → assembly → component → board.
- 131000PMO1: Program Management at a high-level node with no deeper structure.

By enforcing this format:

- Control account codes are always **readable and sortable**.
- You can easily **add new CAs** at any intersection (ENG2, PMO2, etc.).
- Every code carries **meaning about its WBS, OBS, and unique role** - without relying on external lookup tables.

Work Packages: Metadata-Driven, Modular, and Flexible

Having established a structured foundation for WBS, OBS, and control accounts, we can now implement a **clean, compact, and metadata-rich naming scheme** for work packages.

IRDM simplifies work package coding by treating the **control account code as the parent**, and appending a structured metadata suffix using a decimal separator:

Format:
[Control Account Code].[Metadata Code]

In our example program, each metadata code consists of four fixed characters, where **each position carries defined meaning**:

Position	Definition	Example Value
1	**Cost Element** (L = Labor, M = Material, T = Travel, etc.)	L, M, T
2	**CLIN** (Contract Line Item Number)	1 = CLIN1, 2 = CLIN2
3	**Phase** (Lifecycle stage)	A = Design, B = Test, C = Production Prep, D = Production
4	**Index** (Sequence number)	1, 2, 3, etc.

Table 4-Work Package Metadata to Code Mapping

Examples:

- 111SPC1.L1A1 → Labor for CLIN 1, Design Phase, first WP
- 111SPC1.L1B2 → Labor for CLIN 1, Test Phase, second WP

This format gives us several advantages:

- **No ambiguity in code meaning** - every character has a defined purpose.
- **Optimized for reporting tools** - sort, filter, and parse with ease in Excel, Power BI, and SharePoint.

- **Scalable and flexible** - the structure anticipates program evolution. The **index position** enables the addition of new work packages without disruption due to engineering changes, contract modifications, or rebaselining.
- **Extensible by design** - in our example, we've used four metadata positions (cost element, CLIN, phase, and index), but the schema can accommodate more. Need to add recurring/nonrecurring designation? EV method? Functional group? Simply assign meaning to new positions - without changing the structure or breaking existing reports.

The key takeaway: **IRDM work package codes adapt to program complexity** without multiplying control accounts, fragmenting the WBS, or relying on fragile external mapping tools. Your structure becomes self-aware, self-validating, and system-friendly.

Bringing It All Together: Control Accounts with IRDM Naming Applied

We began this journey with the foundational control account diagram - where scope (WBS) intersects responsibility (OBS). Now, let's return to that same structure and apply everything we've built through IRDM.

This time, the control account codes follow our naming convention:

- **WBS lineage** is embedded by flattening the numeric structure into fixed-position integers.
- **OBS responsibility** is encoded with a three-letter, uniquely identifiable string.
- **The sequence number** gives us flexibility for planning refinements or scope additions.

The result is a structure that's not only compliant and clean, but also **decodable at a glance** - by analysts, managers, and reporting systems alike.

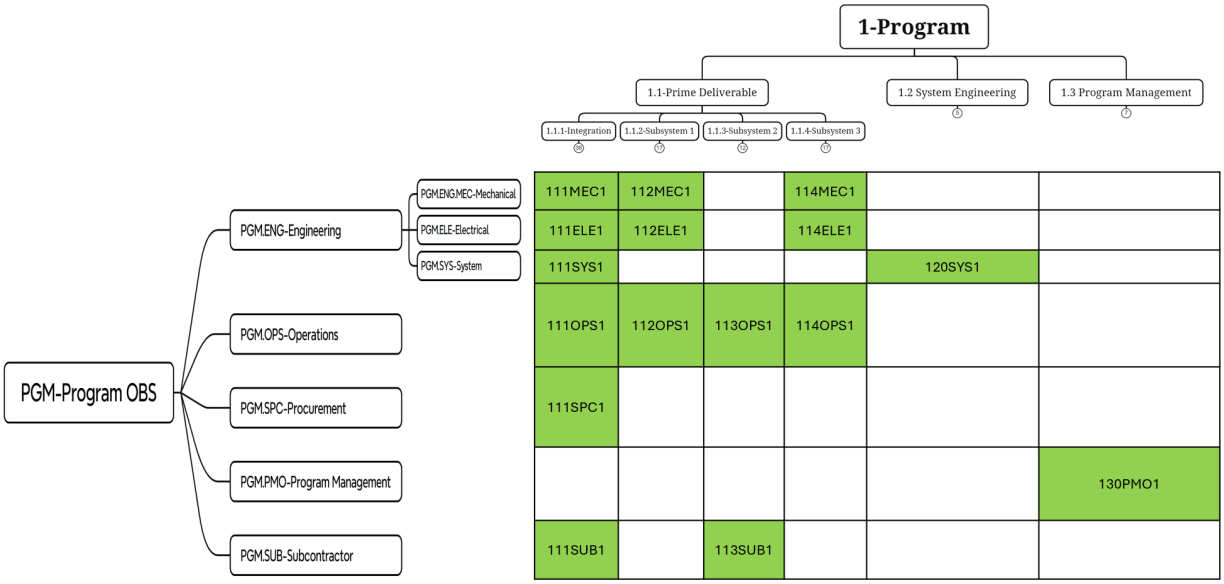


Figure 19-IRDM-enhanced control account view: Same intersections, now with structured naming. Codes reflect WBS ancestry, OBS responsibility, and sequencing - preserving clarity and enabling scalable integration

Automated Setup: Turning Metadata Into Work Package Codes

With our naming rules and metadata structure in place, IRDM allows us to move beyond manual code assignment and into **automated generation**. By building a simple table of all relevant combinations - **WBS, OBS, CLIN, Phase, and Cost Element** - we can rapidly generate a complete, valid, and structured list of work packages with meaningful names and consistent codes.

This table becomes the starting point for:

- **Work package definition**
- **Schedule integration**
- **Charge number mapping**
- **Metadata-driven report filters**

Each line represents a traceable, measurable unit of work. Once this dataset is built, we can use formulas in Excel or logic in Power Platform tools to automatically generate control account references, metadata flags, and even recommended resource pools. And because each work package is tied to its **control account parent**, the entire structure remains stable - even as complexity grows.

What used to take hours or days to configure manually can now be done in **minutes with high reliability** - setting the stage for scalable governance, analysis, and change control.

By combining all integration dimensions into a single table, we create a metadata model for building our work package structure.

Name	WBS	OBS	CLIN	Phase	Cost Element	Control Account	Work Package
Integration Mechanical Design	1.1.1	PGM.ENG.MEC	CLIN1	DES	L	111MEC1	111MEC1.L1A1
Integration Mechanical Test	1.1.1	PGM.ENG.MEC	CLIN1	TST	L	111MEC1	111MEC1.L1B1
Integration Mechanical Production Prep	1.1.1	PGM.ENG.MEC	CLIN1	PRR	L	111MEC1	111MEC1.L1C1
Integration Electrical Design	1.1.1	PGM.ENG.ELE	CLIN1	DES	L	111ELE1	111ELE1.L1A1
Integration Electrical Test	1.1.1	PGM.ENG.ELE	CLIN1	TST	L	111ELE1	111ELE1.L1B1
Integration Electrical Production Prep	1.1.1	PGM.ENG.ELE	CLIN1	PRR	L	111ELE1	111ELE1.L1C1
Integration Systems Design	1.1.1	PGM.ENG.SYS	CLIN1	DES	L	111SYS1	111SYS1.L1A1
Integration Systems Test	1.1.1	PGM.ENG.SYS	CLIN1	TST	L	111SYS1	111SYS1.L1B1
Integration Systems Production Prep	1.1.1	PGM.ENG.SYS	CLIN1	PRR	L	111SYS1	111SYS1.L1C1
Integration Systems Design Travel	1.1.1	PGM.ENG.SYS	CLIN4	DES	T	111SYS1	111SYS1.T4A1
Integration Systems Test Travel	1.1.1	PGM.ENG.SYS	CLIN4	TST	T	111SYS1	111SYS1.T4B1
Integration Systems Production Prep Travel	1.1.1	PGM.ENG.SYS	CLIN4	PRR	T	111SYS1	111SYS1.T4C1
Integration Operations Planning Design	1.1.1	PGM.OPS	CLIN1	DES	L	111OPS1	111OPS1.L1A1
Integration Operations Planning Test	1.1.1	PGM.OPS	CLIN1	TST	L	111OPS1	111OPS1.L1B1
Integration Operations Planning Production Prep	1.1.1	PGM.OPS	CLIN1	PRR	L	111OPS1	111OPS1.L1C1
Integration Operations Production Labor	1.1.1	PGM.OPS	CLIN2	PRD	L	111OPS1	111OPS1.L2D1
Integration Procurement Planning Design	1.1.1	PGM.SPC	CLIN1	DES	L	111SPC1	111SPC1.L1A1
Integration Procurement Planning Test	1.1.1	PGM.SPC	CLIN1	TST	L	111SPC1	111SPC1.L1B1
Integration Procurement Planning Production Prep	1.1.1	PGM.SPC	CLIN1	PRR	L	111SPC1	111SPC1.L1C1
Integration Procurement Production Labor	1.1.1	PGM.SPC	CLIN2	PRD	L	111SPC1	111SPC1.L2D1
Integration Production Material	1.1.1	PGM.SPC	CLIN3	PRD	M	111SPC1	111SPC1.M3D1
Integration Subcontractor Design	1.1.1	PGM.SUB	CLIN1	DES	S	111SUB1	111SUB1.S1A1
Integration Subcontractor Test	1.1.1	PGM.SUB	CLIN1	TST	S	111SUB1	111SUB1.S1B1
Integration Subcontractor Production Prep	1.1.1	PGM.SUB	CLIN1	PRR	S	111SUB1	111SUB1.S1C1

Table 5-Code Generation Supported by Automation

This table drives automation, clarity, and repeatability - enabling structure to scale without bloating control accounts.

From Metadata to Codes: IRDM in Excel

IRDM doesn't just work conceptually - it works in practice, and you don't need custom software to use it. With a structured metadata table, work package codes and control accounts can be **auto-generated using standard Excel formulas**.

Structured metadata enables automation. With simple Excel formulas, work package and control account codes can be generated at scale. For example, in the sample metadata table, we generate codes using the following formulas:

- **Control Account Code**
  =LEFT(B2,1) & MID(B2,3,1) & MID(B2,5,1) & RIGHT(C2,3) & "1"
  This flattens the WBS (in cell B2), extracts the OBS (in C2), and appends a sequence index.
- **Work Package Code**
  =H2 & "." & F2 & RIGHT(D2,1) & SWITCH(E2,"DES","A","TST","B","PRR","C","PRD","D") & "1"

This pulls the control account code (in H2), appends cost element (F2), CLIN (D2), and phase (E2) - then assigns an index.

With this simple setup:

- New work packages are coded **instantly and consistently**.
- Analysts can **add or update rows** without recoding anything manually.
- **Metadata changes flow directly into reports** through formulas or data model refreshes.

IRDM's structure is not only **human-readable** - it's **machine-friendly**, empowering rapid deployment at scale with common tools. IRDM's predictability makes this possible. Programs gain flexibility without sacrificing control or compliance.

Seeing the Metadata in the Code

The beauty of the IRDM naming system is that once established, it **eliminates the need for parallel metadata tables**. The structure itself tells the story. Each code - whether a control account or a work package - is **self-describing**, with clear lineage, responsibility, and integration context embedded in the string.

In the table below, we've removed the metadata columns entirely. What's left? Just the code and the name - and that's all you need. A properly structured IRDM code provides everything required for reporting, filtering, and analysis.

Let's decode an example:

- **Code**: 111SPC1.L2D1
- **Translation**:
    - **111** → WBS = 1.1.1
    - **SPC** → OBS = PGM.SPC (Procurement)
    - **1** → First control account at this WBS/OBS intersection
    - **.L2D1**:
        - **L** = Labor (Cost Element)
        - **2** = CLIN 2
        - **D** = Phase D (Production)
        - **1** = First WP with this metadata set

So, 111SPC1.L2D1 tells us this is a **labor work package**, supporting **CLIN 2**, in the **Production phase**, under the **Procurement control account** aligned to **WBS 1.1.1** - and it's the first such package.

No extra lookup tables. No ambiguity. No wasted time.

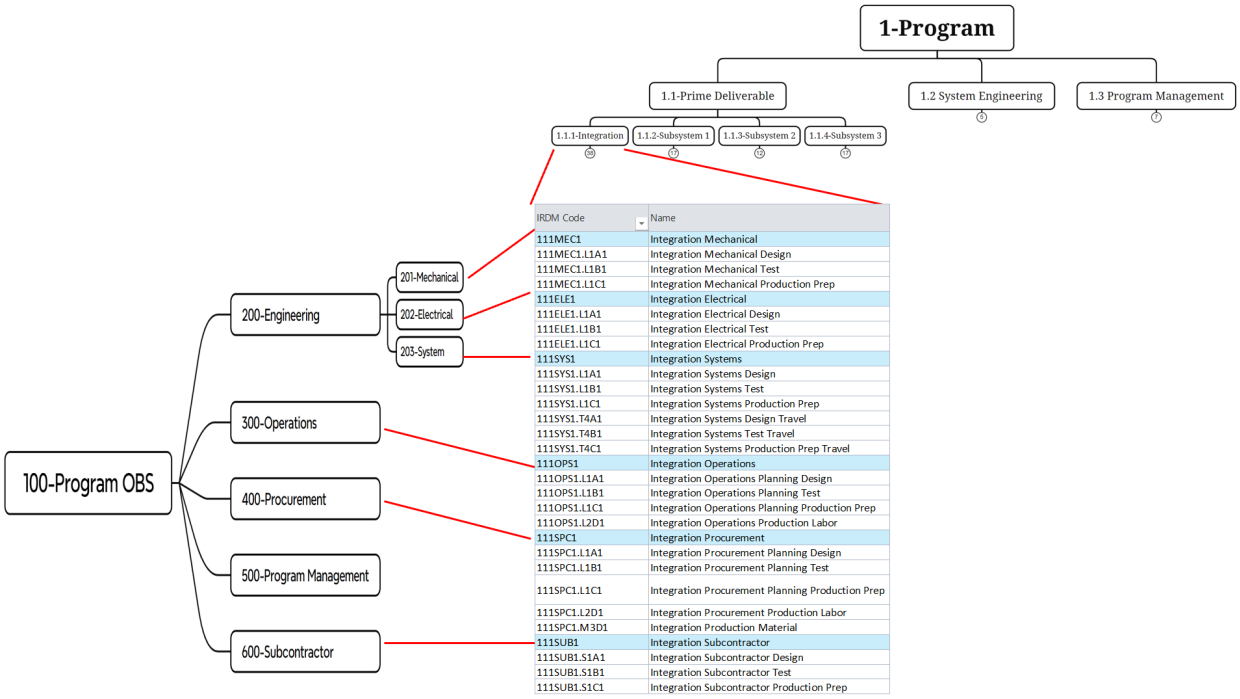


Figure 20-Control Accounts and Work Packages with IRDM Naming Applied

Step 4 - Validate and Review Program Structure

We have a strong foundation for project controls established. Our common structure codes and names all make sense, our work packages and control accounts honor the integration rules we established with our diagram from step 1, and we're almost ready to execute. Now it's time to validate. This critical step allows us to confirm that our proposed structure still aligns with the management approach of the program team, and our reports will meet all data visibility requirements.

Consider our original control account 111SPC1 - responsible for procurement planning in design, test, and production prep, and later for executing procurement labor and managing material acquisition in production. At first glance, this appears manageable. However, upon review with our program and supply chain leadership, we learn that responsibility for procurement labor and material will shift to a different individual once production begins. This change in Control Account Manager (CAM) ownership signals the need for a structural adjustment.

111SPC1	Integration Procurement
111SPC1.L1A1	Integration Procurement Planning Design
111SPC1.L1B1	Integration Procurement Planning Test
111SPC1.L1C1	Integration Procurement Planning Production Prep
111SPC1.L2D1	Integration Procurement Production Labor
111SPC1.M3D1	Integration Production Material

Table 6-Initial Control Account and Work Package Plan for Procurement

Rather than revising the WBS or reassigning massive portions of scope, we simply add 111SPC2 and recode the relevant work packages. All metadata remains intact, and our smart coding conventions ensure continuity. No baseline disruption, no relinking of dependent structures - just a clean, traceable reallocation.

111SPC1	Integration Procurement Pre-Production
111SPC1.L1A1	Integration Procurement Planning Design
111SPC1.L1B1	Integration Procurement Planning Test
111SPC1.L1C1	Integration Procurement Planning Production Prep
111SPC2	Integration Procurement Production
111SPC2.L2D1	Integration Procurement Production Labor
111SPC2.M3D1	Integration Production Material

Table 7-First Revision of Procurement Control Account Mapping for CAM Ownership

Next, we apply our EVM rules. Our EVMS system description disallows combining Level of Effort (LOE) and discrete work in the same control account. Since 111SPC2 contains both, we split once more, introducing 111SPC3. Again, this change is lightweight and reversible. Our system was built to expect change.

111SPC1	Integration Procurement Pre-Production
111SPC1.L1A1	Integration Procurement Planning Design
111SPC1.L1B1	Integration Procurement Planning Test
111SPC1.L1C1	Integration Procurement Planning Production Prep
111SPC2	Integration Procurement Production Labor
111SPC2.L2D1	Integration Procurement Production Labor
111SPC3	Integration Procurement Production Material
111SPC3.M3D1	Integration Production Material

Table 8-Second Revision of Procurement Control Account Mapping for EVM Rules

IRDM's Built-In Resilience

This iterative process highlights a key strength of IRDM:

- **Predictable, minimal-impact changes**: Structural adjustments are handled without cascading rework.
- **Preserved metadata and traceability**: No information is lost or obscured during changes.
- **Compliance alignment**: Rules from EIA-748D, NDIA Intent Guide, and internal EVMS system descriptions can be followed without structural strain.
- **Team collaboration and refinement**: Changes are discovered and validated in partnership with stakeholders, reinforcing shared ownership and accountability.

By planning for change, IRDM enables us to remain compliant, adaptive, and transparent. It shifts the focus from rebuilding structure to refining performance - empowering programs to evolve while maintaining integrity.

Comparing the Hybrid WBS Approach to IRDM

This example demonstrates a core truth in modern project controls: complexity is unavoidable, but **how we manage that complexity determines whether a system remains functional or becomes fragile**.

The **Hybrid WBS approach**, though common, relies on embedding contract structure, programmatic visibility, and reporting requirements directly into the WBS and control account hierarchy. As we saw, this leads to:

- Rapid **growth in control accounts** - from 6 to 23 in a simple example
- Loss of clear **ownership and accountability**
- **Structural rigidity**, making every contract mod or phase change a reengineering effort
- A system built for reporting, not for control

In contrast, applying IRDM preserved structural integrity while delivering **better visibility and greater flexibility**:

- The WBS remained product-oriented and compliant
- Control accounts were held at 8, adding only two new control accounts for management driven reasons and **focused purely on management control**
- All integration was handled at the **work package level**, using metadata and naming conventions
- The structure supported multidimensional reporting without inflating process overhead

The takeaway is simple: **structure should support control, not absorb complexity**. IRDM accomplishes that by assigning each dimension to its rightful place, then connecting them through deliberate, traceable integration points. The result is a leaner, smarter, more scalable system - built for execution, not reconciliation.

Feature	Hybrid WBS	IRDM
CLIN Integration	Via WBS levels	Via metadata in WP
Phase Tracking	Requires additional CA layers	Metadata flags
Control Account Count	Inflates exponentially	Remains lean
Traceability	Complex joins, manual work	Formula-based, auto-coded
Change Resilience	Rigid, breaks easily	Modular, easily updated

Table 9-Feature Comparison between Hybrid WBS and IRDM

While our example illustrates the application of IRDM in a simplified environment, the method was shaped in response to real-world complexity. In practice, IRDM was developed and refined under the pressure of a multi-CLIN, multi-funding-stream DoD program where structural failure was not an option. The following summary connects that experience to the principles applied here.

Real-World Application: IRDM at Scale

The IRDM approach wasn't built in a vacuum. It was forged in the fire of an $800+ million, 7-year Department of Defense IDIQ program - one with development and production scope, required EIA-748 EVMS compliance, and over 40 CLINs spread across multiple delivery orders and fiscal years. The program required strict compliance with MIL-STD-881F and had additional structuring mandates from multiple Program Executive Offices. Most critically, it needed to support three overlapping funding streams, while providing visibility into department-specific bidding structures and cost collection at the bid task level.

Traditional organizing models were insufficient. The control account hierarchy was under constant strain - managing formal change, internal variance analysis, and external visibility across more than three contract modifications per month. Trying to build that kind of system using a hybrid WBS would have meant collapsing under the weight of structure alone.

The IRDM framework enabled this program to pass its DCMA EVMS compliance review on the first attempt. More importantly, it allowed the team to manage over 1,000 work packages across a highly dynamic contract landscape without structural fragility. The diagram below illustrates how the program mapped Common, Contract, Accounting, and Program-Specific structures while maintaining a stable, lean control account base:

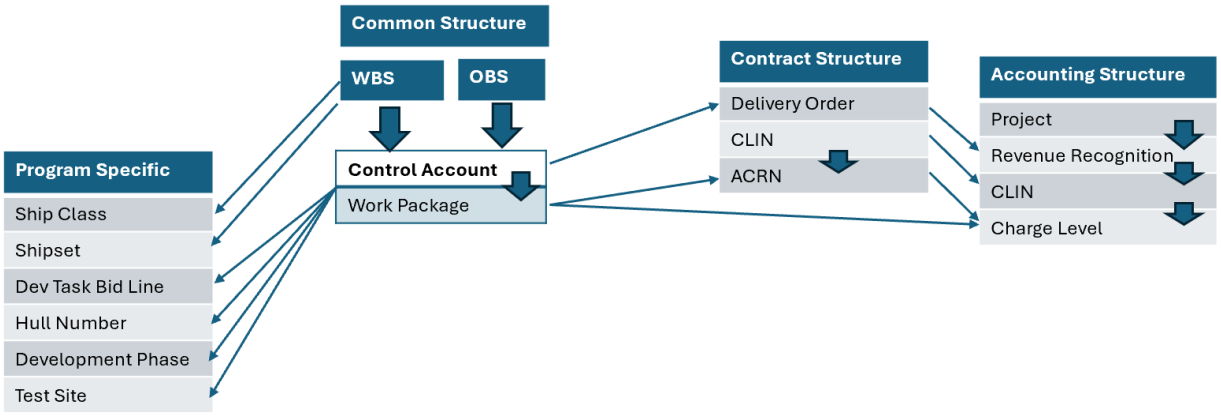


Figure 21-Applying IRDM to a Complex Contract

Over time, this framework didn't just hold - it scaled. With each contract modification, delivery order, and funding increment, the program added new metadata and work packages without breaking its hierarchy or inflating its control account count unnecessarily.

Good structure doesn't just support compliance - it unlocks insight. With IRDM in place, every data point in the system is traceable, sortable, and meaningful. To demonstrate this,

we built a Power BI report that leverages structured metadata and naming logic to deliver immediate management visibility across dimensions - without modifying the structure.

From Structure to Insight: Interactive Reporting with IRDM

To demonstrate the practical power of IRDM, we developed a working Power BI prototype. This model consumes structured metadata to drive dynamic dashboards that respond instantly to slicing by WBS, OBS, CLIN, phase, cost element, and more. There's no restructuring, no manual remapping, and no ambiguity.

Clean structure unlocks clean reporting. These Power BI screenshots show real-time filtering by CLIN, phase, EV method, and function.

The first report image shows the data filtered by the Design phase. The report immediately identifies the EV Method mix in design, that design is $19.5 million in BAC, and that it is 66% labor and 33% subcontracted by dollar value. It also shows the budget splits by WBS and OBS.

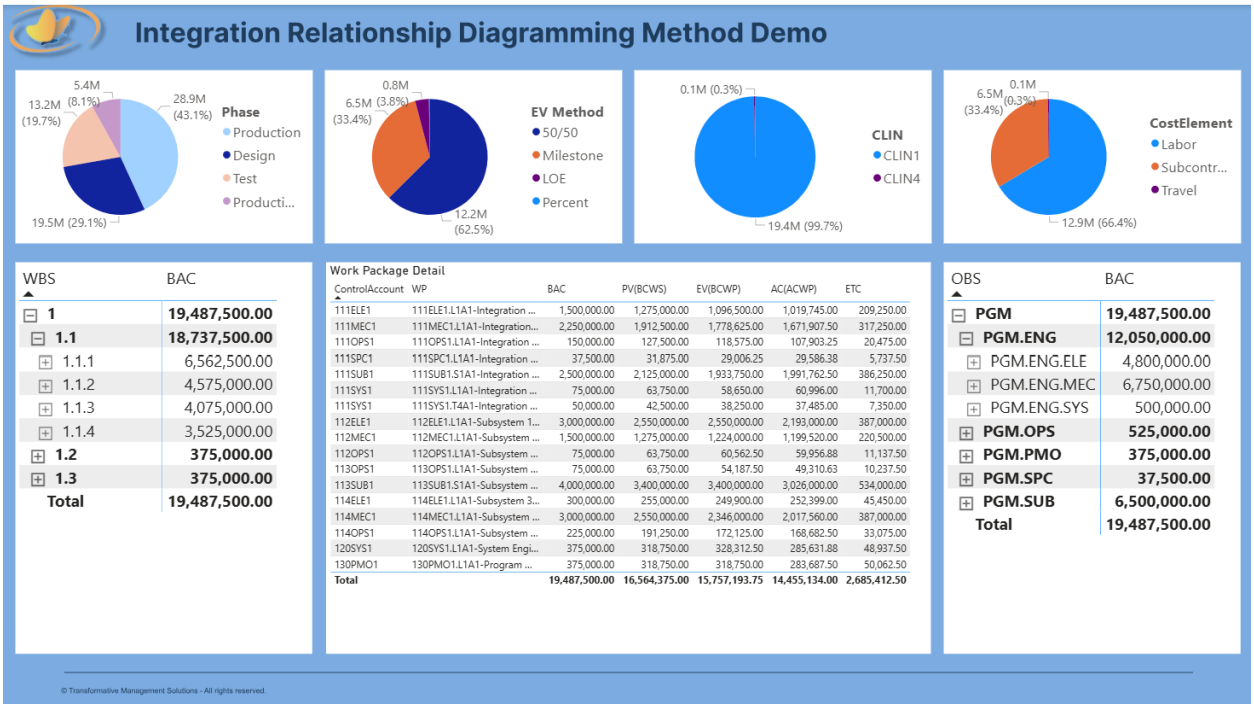


Figure 22-PowerBI Screenshot, Design Phase Filter Applied

The next image is generated by selecting a different filter mix. In this image, the data is viewed by Production. We can easily see the shift in EV Methods, CLIN allocations, and cost element mix, as well as the shift from engineering to operations and supply chain in the OBS.

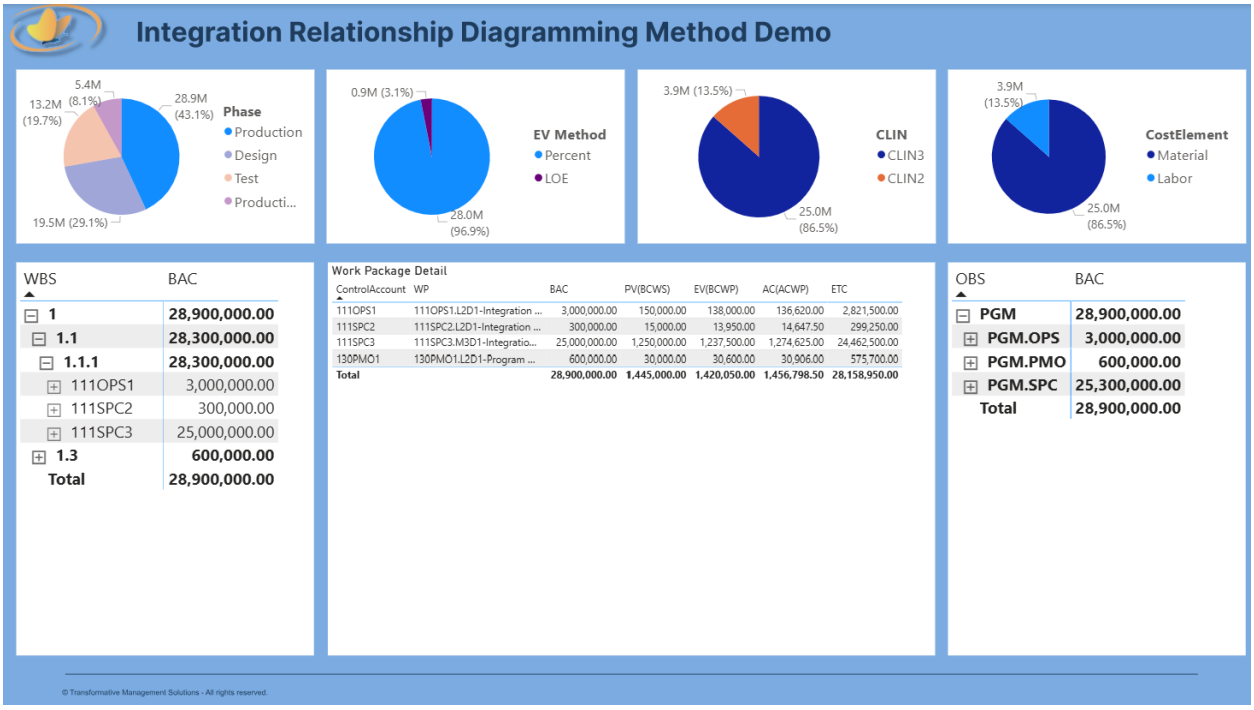


Figure 23-PowerBI Screenshot, Production Phase Filter

The final image demonstrates a complex cascading filter example, where CLIN1, Labor, and Test Phase are all selected. The metadata assignments provide simple and fast access to data combinations that directly address critical management questions.

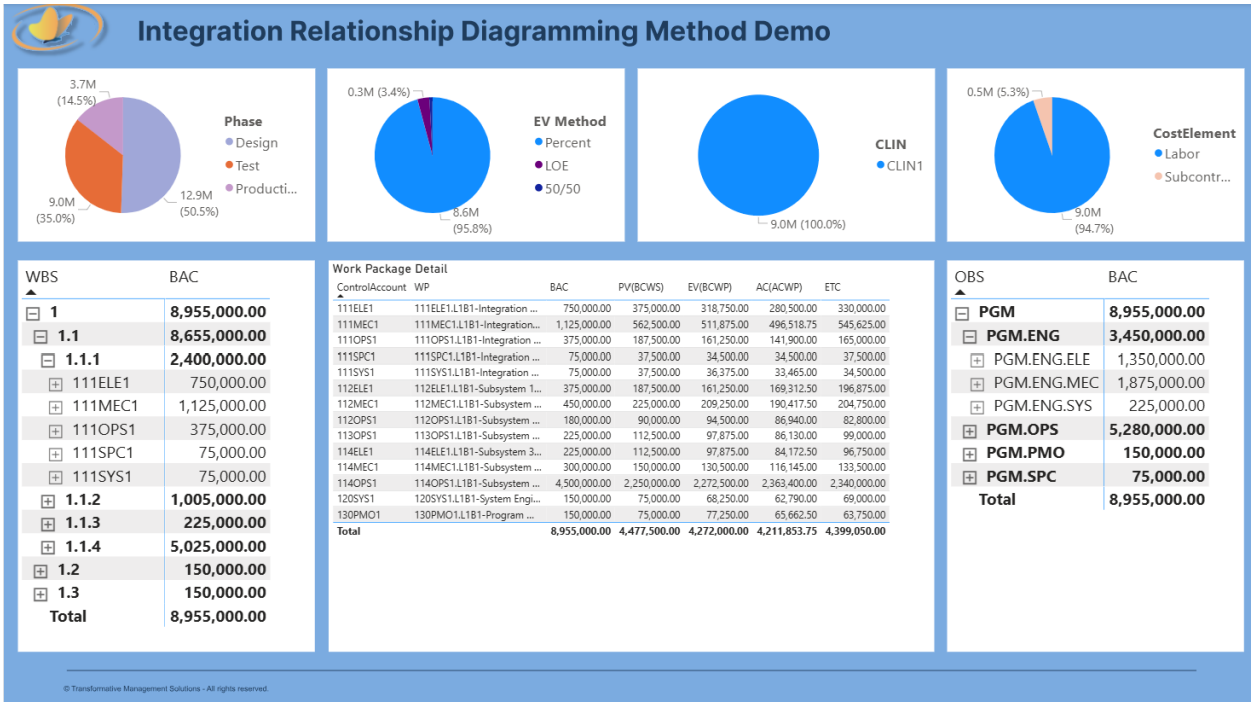


Figure 24-PowerBI Screenshot, Combination CLIN1, Labor, Test Phase Filters Applied

These dashboards are not static visualizations - they're dynamically populated based on structured codes and metadata from the IRDM method. This allows teams to answer complex questions, track scope ownership, and respond to audits or customer inquiries without scrambling through spreadsheets or revisiting structure decisions.

Conclusion: A Smarter Path to Integrated Control

The structural challenges demonstrated in this article are not theoretical - they are embedded in the daily reality of every complex, high-stakes program. Project teams are being asked to manage more complexity, deliver more visibility, and adapt to faster cycles of change - all within control systems that were never designed to scale that way.

The traditional hybrid WBS model, once considered a practical compromise, has reached its limit. By attempting to force multidimensional requirements into a two-dimensional structure, it generates unnecessary control accounts, burdens execution teams with process overhead, and obscures the very insight it was intended to support.

The Integration Relationship Diagramming Method (IRDM) offers a proven, standards-aligned alternative. It clarifies the role of each structural element. It defines how data dimensions should be mapped - not embedded. And it elevates the work package as the natural integration point, where cost, schedule, scope, and performance truly converge.

More than a theoretical framework, IRDM has been field-tested under real-world conditions - supporting compliance, traceability, and control in one of the most complex contract environments in the industry. It works with standard tools. It aligns with existing guidance. And it makes change manageable.

For organizations ready to move beyond patchwork structures and toward scalable, resilient project control systems, IRDM offers not just a method - but a blueprint for long-term structural integrity.

Glossary

Term	Definition
WBS (Work Breakdown Structure)	A hierarchical, product-oriented breakdown of the total work scope of a program.
OBS (Organizational Breakdown Structure)	A structure that defines the organizational entities responsible for executing the work defined in the WBS.
Control Account (CA)	Management control point established at leaf node intersections of WBS and OBS. The primary management and process control point in an EVMS.
Work Package (WP)	The lowest-level planning element where scope, cost, schedule, and earned value are integrated. Work packages carry metadata for reporting and traceability.
Planning Package	A time-phased budget placeholder below a control account for future work scope that cannot yet be planned in detail. Converted into work packages when ready.
CLIN (Contract Line Item Number)	A contractual element used to identify and track deliverables, funding, or invoicing categories within a contract.
SLIN (Sub Line Item Number)	A subordinate line item under a CLIN used to break down deliverables or funding for additional traceability.
ACRN (Accounting Classification Reference Number)	A funding code used by the government to track and manage appropriated funds by source.
EV Method (Earned Value Method)	The technique used to measure progress and earned value on a work package (e.g., LOE, 50/50, Milestone, % Complete).
LOE (Level of Effort)	An EV method used for effort-based work that cannot be discretely measured; earned value is spread evenly over time.

Term	Definition
Discrete Work	Work that has measurable output or deliverables, planned and tracked using objective EV methods.
IPMDAR (Integrated Program Management Data and Analysis Report)	The DoD's required digital reporting format for EVM data submissions, replacing traditional CPRs.
CSDR (Cost and Software Data Reporting)	A formal DoD cost reporting requirement used to collect cost, software, and functional data across programs.
CAM (Control Account Manager)	The individual responsible for planning, executing, and controlling the work within a control account.
Metadata	Data that defines attributes of a structural element, such as phase, cost element, CLIN, EV method, or function - used for filtering, reporting, and traceability.
Hybrid WBS	A structure that integrates all visibility requirements (e.g., CLINs, phases) directly into the WBS hierarchy.

References

1. EIA-748-D (Earned Value Management Systems)

EIA-748-D Earned Value Management Systems. Arlington, VA: SAE International, 2018. Developed by the Government Electronics and Information Technology Association (GEIA) and maintained by SAE International.

2. DoD Earned Value Management Implementation Guide (EVMIG)

U.S. Department of Defense Earned Value Management Implementation Guide (EVMIG). Washington, DC: Office of the Under Secretary of Defense (Acquisition and Sustainment), March 2019.
Available via acq.osd.mil

3. NDIA Intent Guide (EVMS Intent Guide)

NDIA Integrated Program Management Division (IPMD) EVMS Intent Guide, Revision 1. Arlington, VA: National Defense Industrial Association, January 2018.

4. Planning and Scheduling Excellence Guide (PASEG)

Planning and Scheduling Excellence Guide (PASEG), Version 5.0. Arlington, VA: National Defense Industrial Association Integrated Program Management Division (NDIA-IPMD), February 2020.

5. MIL-STD-881F (Work Breakdown Structures for Defense Material Items)

MIL-STD-881F: Department of Defense Standard Practice – Work Breakdown Structures for Defense Materiel Items. Washington, DC: Department of Defense, October 2018.

6. IP2M METRR (Maturity and Environment Total Risk Rating)

Gibson, G. E. Jr., El Asmar, M., Sanboskani, H., & Aramali, V. (2022). *Implementing the Integrated Project/Program Management (IP2M) Maturity and Environment Total Risk Rating (METRR) Using EVMS in a Team Environment (Report No. 6)*. School of Sustainable Engineering and the Built Environment, Ira A. Fulton Schools of Engineering, Arizona State University.