CA Gen

Change Management

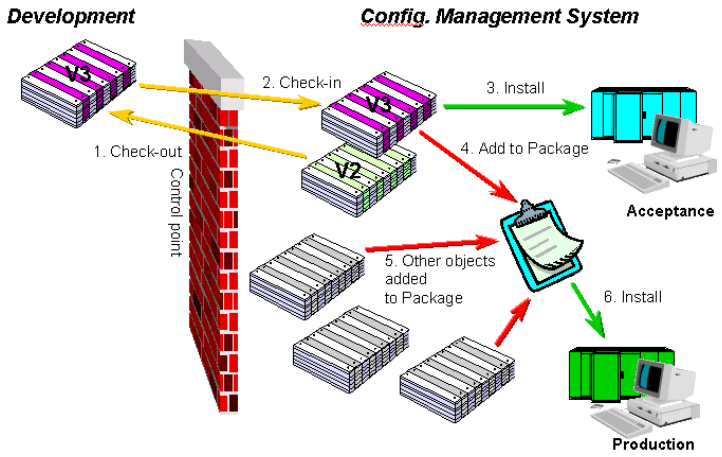
Introduction

Reduced software product life cycles and increasingly diverse requirements are leading companies to review and improve their software development processes. These processes and supporting infrastructure can improve productivity and ensure the product meets customers' requirements and quality expectations.

Software Configuration Management solutions establish and maintain the integrity of the deliverables of a development project throughout its life cycle. Software Configuration Management involves identifying the configuration of the software, systematically controlling changes to that configuration, and maintaining the integrity and traceability throughout the application development life cycle.

Traditional Approach

Before the introduction of model based development tools, the primary controllable item was program source code. Change and configuration management (CCM) tools were used to control changes to source code, to ensure that changes were accurately installed in the target system and providing audit traceability. A typical approach to software configuration management for traditional hand-coded systems is illustrated in the diagram below:

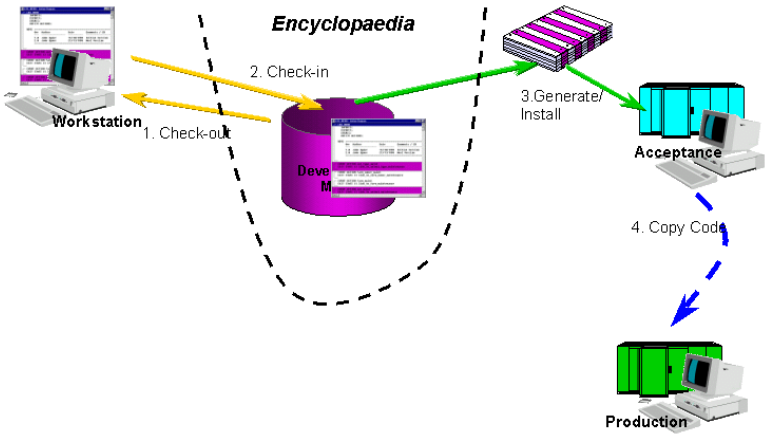


1. The developer checks one or more programs out of the CCM tool to a development library, and applies the necessary changes.
2. When the changes have been tested in the development environment, the programs are checked back into the CCM tool, which automatically assigns a new version number (V3 in diagram).
3. Build facilities in the CCM tool are used to install the changes into the test environment.
4. The changed module is added to a 'package' of changed modules.
5. Other modules can be added to the package.
6. Once all of the changes have been tested, the package of changes is installed into the production environment. The CCM tool ensures that all of the processes required to correctly move the changes to the target environment are performed.

Model Based Development

With CA Gen, the true 'source' of the system is no longer the generated code, but the model(s) from which the code is generated. Changes to the application are applied to the CA Gen models, and the program source code is re-generated. The diagram below illustrates a typical sequence:

1. A 'subset' is scoped and checked out from the CA Gen Encyclopedia to the Workstation to allow the CA Gen objects to be changed.
2. Once the changes have been made and tested on the workstation, the subset is checked back into the Encyclopedia.

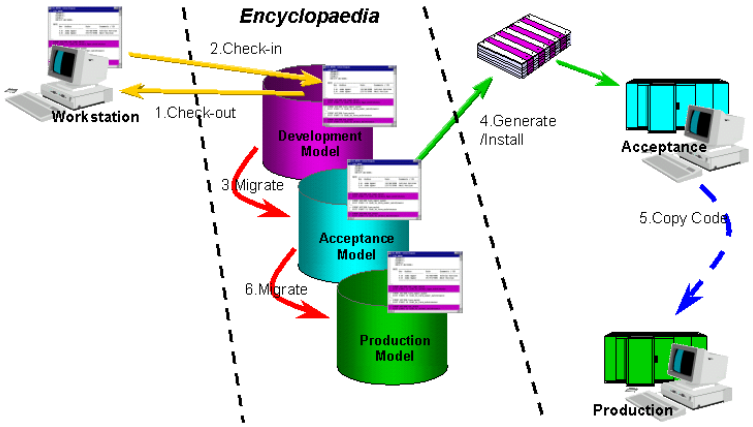


3. The source code is regenerated from the CA Gen Encyclopedia and installed into a test environment.
4. Upon completion of testing, the changed executables are copied to the production environment in order to launch the changes into production.

The most widely adopted approach to version controlling CA Gen objects is to maintain one model that equates to each environment in the development cycle. This ensures that changes applied to the first (development) model do not overwrite the current versions implemented in the later environments. This is illustrated in the diagram below. In this example, the project is supporting a Development, Acceptance Test and Production environment; hence, there are three corresponding CA Gen models.

When a package of changes is ready to be promoted from one environment to the next, the objects are promoted using the CA Gen Object Migration facility.

The source code can then be generated from the model and installed into the target environment.



CA Gen Change Management Strategy

The use of CA Gen does not eliminate the need for sophisticated CCM tools. Rather, the greater speed and responsiveness of a CA Gen based development environment, coupled with greater sharing and re-usability of components, increases the need for well defined and supported configuration management procedures.

To ensure quality and increase productivity during the process of promoting changes from development through to production, sophisticated change and configuration management tools are needed that address the modeling environment and not just the generated code.

GuardIEn provides comprehensive version control, model management and configuration management tools specifically designed for CA Gen application development projects.

IET's GuardIEn Change and Configuration Management tool has been used by many CA Gen users to successfully and efficiently manage their CA Gen models and applications.

GuardIEn is used to manage changes that occur within the CA Gen models. It provides facilities for versioning CA Gen objects and for automating the key CA Gen processes of object migration, impact analysis and system implementation. This option ensures that changes to the CA Gen models are correctly managed and translated to generated source code in a controlled a verifiable manner.

GuardIEn can also synchronize with Enterprise Change and Configuration Management (CCM) tools (like CA's CA Endevor Change Manager and CA Harvest Change Manager, or Serena's ChangeMan ZMF and Dimensions), automatically transferring the generated code into the CCM tool.

GuardIEn Overview

Designed specifically for CA Gen, GuardIEn understands how to version control CA Gen objects, migrate between models, automate impact analysis, execute the CA Gen code generators and automate the installation of CA Gen generated code.

GuardIEn does not just control the movement of code into production, it also offers a high degree of support during the development stages of a project. It can automate model management for a wide variety of model management approaches. It also offers sophisticated impact analysis and reporting capabilities, and can automate the code generation and installation steps for the development environments.

Change Control

All automated processes depend on the quality of the input data, and the automation of the change management processes is no different. Without a clear definition of the changes that have been applied to the models, it is almost impossible to perform an incremental update to the target system. This can result in the need to re-generate and re-

implement the entire application. The clear specification and management of changes to the application is therefore an essential pre-requisite for efficient and rapid implementation of the changes, and automation of the implementation processes.

With traditional CCM approaches, changes to the source code are identified when the code is checked into the CCM tool. With CA Gen, the changes are applied to a subset of the model on the developer's workstation and then uploaded to the encyclopedia. The changed objects are then allocated a new last changed timestamp, but the changed objects are not versioned or associated to any specific change request. It can therefore become extremely difficult to identify which objects need to be promoted when a certain set of change requests have to be promoted.

GuardIEn contains a change control module that enables the central definition of change requests. The unique advantage of using GuardIEn to manage changes to CA Gen models is that GuardIEn will automatically detect new and changed objects at the time the changes are uploaded to the encyclopedia. The changed objects are versioned and associated to the Change Request. This means that developers do not need to maintain separate lists of changed objects, thus eliminating the possibility that a change is forgotten.

Version Control

GuardIEn enables explicit identification of versions. When a developer uploads a changed object to the CA Gen Encyclopedia, GuardIEn detects the new or changed object and ensures that the object is assigned to a unique version. It can automatically 'footprint' a CA Gen action diagram by amending the first NOTE statement to include the version number, date/time of the change, user that made the change, and change request number.

Each new version's progression is managed through the software development cycle. It allows each development team to document their own environments and life-cycles, not only to ensure that they remain consistent with their own working practices, but also enforcing conformance to site-wide standards.

GuardIEn also contains a facility which allows each change to an object to be recorded. This operates at the action diagram statement/property level, thus providing a full history of all changes made to an object over time. You can now see who changed what & when, whereas the CA Gen model will only provide the current object and who made the last change.

Task Management

GuardIEn contains integrated task scheduling tools that manage CA Gen activities as background tasks. Sophisticated software ensures that the overall CA Gen Encyclopedia workload is scheduled according to defined priorities, avoiding encyclopedia and DBMS contention. This maximizes encyclopedia performance and availability of this critical resource. The task management software can also be used for ad-hoc development tasks like downloads, uploads and code generation to ensure that all encyclopedia activities are managed efficiently.

Release Management

An important part of effective configuration management is a clear and easy-to-understand release management policy that facilitates the scheduling of changes to the production system. It should cater to medium- to long-term enhancements as well as short-term fixes that may be required.

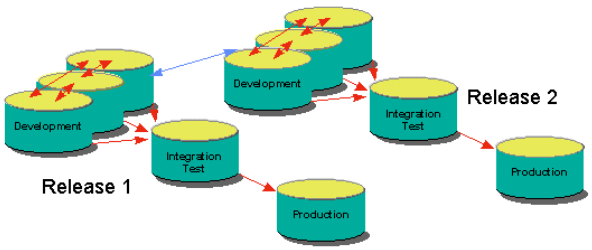
GuardIEn helps projects to define and plan release implementations by providing facilities to document the releases and their content. Multiple system releases can be defined as either independent releases, or linked as a hierarchy of releases, which assists in the tracking of changes between releases.

GuardIEn allows the parallel development of system releases. This enables the team to work on more than one release at the same time. The following diagram illustrates an example of a model architecture for parallel development.

Parallel Development

One of the challenges of parallel development is ensuring that any changes or fixes that are applied to a release are also applied to the subsequent release. This is illustrated in the diagram above by the blue double-headed arrow linking the development models for Releases 1 and 2.

GuardIEn contains special facilities for tracking changes in one release and detecting whether they can be safely migrated to subsequent releases. It uses a concept called 'baselining' to detect whether the object has been changed in the subsequent release, whether it can be migrated, or whether the change has to be manually applied to the next release to avoid losing any changes made in that release.



Impact Analysis

GuardIEn contains facilities to assist in rapid and effective impact analysis. It directly accesses the contents of the models to provide answers to the questions including:

- When was an object last changed in each of the models?
- When was it last generated?
- When was it last migrated?
- What objects use the object, or what objects does it use?

- What objects have been changed but not re-generated?
- Is the current model timestamp the same as the last time it was placed into production?
- To which subsets is the object checked out?
- What change requests have scoped the object?

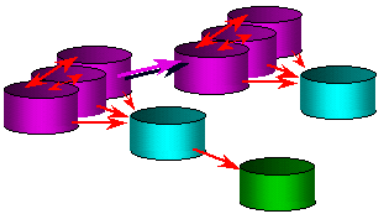
All of these questions and more can be answered very quickly by using the integrated impact analysis features. Because the data is directly extracted out of the CA Gen Encyclopaedia, it is up to date. Moreover, unlike workstation-based reports that operate on a subset of the data, it is complete because it is extracted from the complete model. These reports go beyond single model reporting; they can span multiple models, providing the sort of powerful cross model reports that are required to manage a multiple model development environment.

Model Management

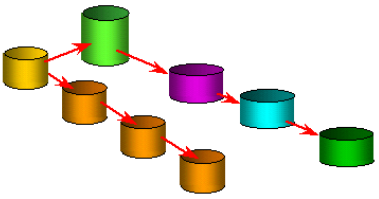
GuardIEn automates many of the checks and tasks necessary to successfully manage multiple CA Gen models. It allows the project to define its model architecture using a rules based approach. The migration rules specify the points in the deliverable life cycle that indicate a migration is required, the pre-conditions and post-conditions for the migration and which model or models should be targeted.

The migration rules provide great flexibility, enabling support for all of the commonly used model architectures. The figure below illustrates some of the typical model architectures supported. Examples include:

- Simultaneous development of more than one system release (parallel development). GuardIEn contains facilities to 'baseline' the objects, to enable the migration of changed objects from one release to the next without overwriting deliverables that have been changed in subsequent release(s).



- Facilities for the distribution and receipt of common objects across projects, including support for Component Based Development model architectures. A 'specification' migrate capability automatically converts an action block to a specification as part of a catalog model migration.



System Updating

GuardIEn automates the implementation of changed objects into a target environment, including the updating of the executable system. The steps required to implement a system will vary between organizations.

Typical steps include:

- Back up the model prior to implementing any changes.
- Migrate new and changed deliverables to the controlled model.
- Perform impact analysis on changed objects to determine what modules require regeneration and installation.
- Execute CA Gen code generation tools to generate source code.
- Install source code (compile, link-edit, etc.).

GuardIEn executes the steps using a series of linked background tasks. It then checks the results, and if the step was successful, automatically moves on to the next step. This allows the update to be performed overnight without any intervention. Contrast this with a manual process, whereby each step must be performed individually, often split over several days to allow for example the completion of one step before defining and executing the next step.

External Objects

Whilst the Gen models contain a large percentage of the project's deliverables, there is often a need to manage and control 'external' objects, i.e. source code, bitmaps, OLE control files, icons, JCL, DDL, documentation and other project artefacts that cannot be stored within the Gen models. GuardIEn's XOS (External Object Support) module provides facilities for versioning, storing and managing these external objects. Uniquely it also provides integrated impact analysis tools that enable the associations between the Gen and external objects to be controlled. For example, if a Gen server is changed, the external clients that access the server can be easily viewed alongside the Gen clients that also use the server module. GuardIEn system updates can also process external objects so that the promotion of these objects through the software life-cycle is co-ordinated with the Gen model management and code installation steps.

Benefits

Error Free Implementations

Automated system updating facilities of GuardIEn eliminate the errors typically associated with using manual processes. Additionally, many customers report a dramatic reduction in the numbers of system failures following the introduction of GuardIEn.

Faster Implementations

The automation of the implementation tasks can significantly help reduce the overall time required to develop and implement new functionality, and fix problems.

- Integrated impact analysis and audit/history tracking results in quicker identification of the causes of problems.
- Automated migrations, code generations and installations can be scheduled as a series of linked background tasks, rather than manually submitting and checking each step before proceeding to the next step.

Improved Model Management Productivity

By automating tasks like object migration, model compares, checking for common ancestry, impact analysis, and so on, GuardIEn significantly reduces the time and effort required for model management.

Customers experience typical productivity savings of 50% for the effort required to implement changes through the application life-cycle.

Improved Developer Productivity

By providing explicit versioning of CA Gen objects, integrated change control and impact analysis tools, GuardIEn makes it easier for developers to document changes, check that changes are consistent with other changes applied to the application and understand the impact of the changes on related parts of the application.

Additionally, since automated object checking reduces the number of failed migrations, models will be available and will not be locked by repeated migration, and thus developer productivity is increased.

Moreover, automation of the code generation and installation tasks provides a significant saving in the effort and time needed to implement changes.

The use of the integrated subset definition, task scheduling, impact analysis and automated implementation tools typically improve developer productivity by at least 10-15% during the detailed design and construction phases of a project and up to 20% during the maintenance phases.

Standard Working Practices

GuardIEn supports the use of standard working practices that are tailored to suit an organization's or project's desired approach to model management, change management and status reporting. By automating many of the CA Gen model management and implementation processes, GuardIEn allows these key processes to be performed according to a defined standard by staff that are not necessarily CA Gen model management experts. This reduces the dependency on the continuous availability of skilled CA Gen model management staff , though it should be noted that the organization will still require expertise in CA Gen model management.

Compliance

GuardIEn has enabled development projects to achieve compliance with the industry standards in software configuration management, for example, CMMI, Sarbanes-Oxley and ISO 9000.