Information Engineering Technology

DevOps Tool Support for Gen

IET's tools provide automated support for many of the tasks and processes involved in the development and management of Gen applications. GuardIEn has been used by many Gen customers to provide change control, versioning and automated build and deployment for over 30 years. GuardIEn also provides interfaces to commonly used enterprise change and configuration management tools on both z/OS and distributed platforms.

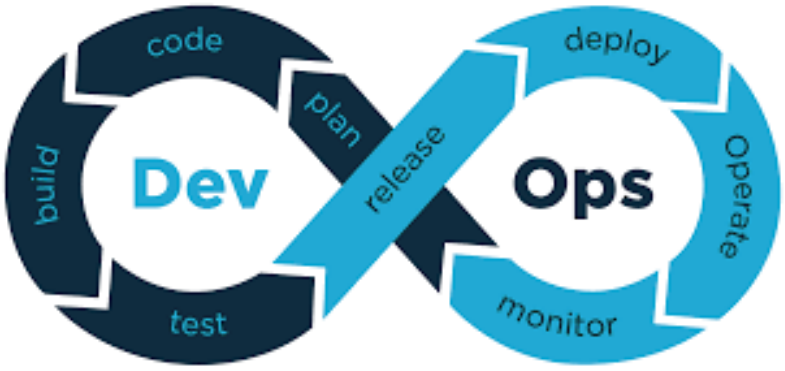
Many organisations are now revising their development practises for greater tool integration under the banner of DevOps. Whilst most users of GuardIEn and other IET tools will already be automating many of the steps in the life-cycle, in the context of DevOps they may also want to consider greater integration between tools. This paper describes the support currently offered by IET tools and opportunities for further tool integrations.

What is DevOps?

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

Whilst there isn't a formal, universally accepted definition of DevOps, the Software Engineering Institute have suggested defining DevOps as "*a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality*".

The graphic below is used to illustrate the continuous cycle of development and operations.



Typical goals for DevOps include:

- Improved deployment frequency
- Faster time to market
- Lower failure rate of new releases
- Shortened lead time between fixes
- Faster mean time to recovery

Automation

A key aspect to achieving these goals is effective automation of the build and release processes, including effective configuration management and version control of the software artefacts. It is beyond the scope of any one tool to automate all of the processes involved in DevOps, so effective automation relies on using multiple tools and products.

A key part of successful DevOps is continuous integration and continuous delivery (CI/CD).

Continuous integration (CI) is a process in which developers and testers collaboratively validate new code, and it automates the code, build and test stages.

Traditionally, developers wrote code and integrated it for testing, sometimes monthly or even less frequently. To avoid the problem of having to fix code that was written weeks or months before, CI uses automation to integrate code continuously (or much more frequently) in the release and deploy stages.

Continuous delivery (CD) is the process of continuously (or frequently) creating releasable artefacts to test that the software product is executable and facilitate testing.

In an ideal world, and with the benefit of an unlimited budget, you could contemplate automating every process and integrating every tool. In the real world and with restricted budgets this is usually not achievable, especially when most enterprises are creating software using a diverse set of software development tools, for multiple platforms and with teams spanning multiple locations.

It is important therefore to focus on the most beneficial automation and tool integration for the individual enterprise's own environment taking into account existing tools, processes, culture and maturity.

Candidate processes for automation should be identified and prioritised, for example by asking:

- How frequently is the process repeated?
- How long does the process take?
- Is the process causing delays?
- How repeatable is the process and is it reliant on the skills and diligence of key individuals?
- Is the process error-prone if it is not automated?

Many of the processes involved in managing and deploying a change to a Gen application are time-consuming and highly reliant on the skills of a few individuals and therefore key candidates for automation.

The Challenge with Gen

The majority of software development approaches and tools involve creating source code that is stored as text files and then compiled/built into an executable. Many tools are available for version control and build automation, and because the source code is file based, standardised tools are able to automate the processes without needing to understand the precise underlying content of the specific language or tool used to develop the source code.

Gen is very different in this regard. The ‘source code’ is not the generated C, Java or COBOL but the model stored in the encyclopaedia, or more specifically, a highly structured and interrelated set of objects, associations and properties that are stored in a proprietary format in a database.

Standardised configuration management and release automation tools do not have direct access to the software artefacts in the Gen models and they are only of use once the 3GL source code has been generated from the model.

The processes involved in managing and building Gen applications are very specific to Gen, for example building applications from multiple models, object migration and analysing the impact of changes to the model in terms of what programs require regeneration. They not only require a very good and in-depth understanding of Gen, but their successful execution is also reliant on a good understanding and documentation of the Gen model and application architecture and contents.

A fundamental part of successful automation of the code, build and test phases of development is source code control with accurate versioning of the changes made. Whilst with Gen you can determine what was last changed, automation of the DevOps processes requires more precise control. Specifically, it is vital to have the ability to associate the changes to individual objects back to the originating change requests so that the scope of a release of the software in terms of changes to the Gen models can be accurately assessed to determine what code needs to be regenerated. Without this you are left with needing to regenerate all of the changes irrespective of whether the changes are completed and ready for release. Lack of true version control can be acceptable for a ‘big bang’ release of all changes with a mass regeneration of code, say once per year, but is not compatible with the desire for frequent and agile releases of smaller packages through the life-cycle, for example with Continuous Integration / Continuous Delivery.

IET Tool Support

IET's suite of tools are focussed on supporting the planning, coding, build and test stages of the DevOps life-cycle. These are summarised in the table below.

DevOps Phase	Tools	Description
Plan	*GuardIEn: Change Requests*	GuardIEn Change Requests are used to document the business requirements. If other tools are used for this purpose, then they can be integrated with GuardIEn CRs.
Code	*Studio Developer*	Studio Developer is an Eclipse based workbench for developing Gen applications with full integration to GuardIEn, VerifIEr and pathvIEw.
	VerifIEr: QA & Code Checking	VerifIEr provides static code analysis for Gen models. Automated checking detects errors and non-compliance with site standards enabling fast and cost effective correction at an early stage of the development process.
	GuardIEn: Version Control	GuardIEn provides version control of Gen objects, providing a complete audit trail of what was changed, when, why (via connections to Change Requests) and by whom. This is the pre-requisite and foundation for build and release automation.
Build	*GuardIEn: System Updating*	GuardIEn System Updating is used to automate the build processes for Gen applications, including interfaces to other enterprise build tools where required.
Test	*pathvIEw: Code Coverage*	pathvIEw provides code coverage testing for Gen applications at the action diagram statement level to measure and improve the quality of manual or automated testing.
	xTrace	xTrace allows you to trace/debug your Gen z/OS applications in batch, CICS or IMS. With powerful breakpoint support and a advanced user interface, it significantly speeds up testing of z/OS applications.
Release	*GuardIEn: System Updating*	GuardIEn System Updating is used to automate the release processes for Gen applications, including interfaces to other enterprise release automation tools where required.

GuardIEn: Configuration Management for Gen

GuardIEn was designed from the ground up specifically and uniquely as a change and configuration management solution for Gen. It automates version control, change management and automated build and release management.

With GuardIEn the architecture of the project is precisely defined in terms of the development life-cycle, model architectures, promotion rules and steps required to automate the build and deployment.

GuardIEn can either be used as a standalone tool or it can be integrated with other products including software configuration management, release automation and change tracking tools.

The key functions in GuardIEn of relevance to DevOps are described below.

Change Requests

The development cycle starts with planning or requirements definition. In GuardIEn a Change Request is defined to identify the business or technical requirement.

Version Control

As the developer develops the changes, new, changed or deleted objects in the Gen model are automatically associated to the GuardIEn Change Request via the GuardIEn Upload Assistant which detects changed objects and establishes new versions for them. GuardIEn versions allow you to see precisely what each separate change involved, for example changes at the action diagram statement level.

Release Packs

A GuardIEn Release Pack is a grouping or package of one or more Change Requests that are to be promoted together, for example as the scope of a new release or update to an application.

System Updates

A GuardIEn System Update is used to automate the build process via the execution of various steps, for example, object migration, impact analysis, code generation, external action block compilation, build and any other process required to convert the Gen and external sources into an executable system.

GuardIEn System Updates can be executed on demand via the GuardIEn client user interface or scheduled to run automatically at certain intervals or when triggered. This forms the basis for automated continuous integration since the GuardIEn SU is scoped to promote all of the objects changed within the scope of the SU via its association to the Change Requests.

Interfaces

Whilst a GuardIEn System Update can automate all of the build and deployment steps for a wide variety of application targets, often an enterprise will use a standard tool for release deployment. To enable Gen to participate in the enterprise-wide standard process, GuardIEn provides interfaces to enable handoff of the build process. Standard interfaces are available for commonly used products like ChangeMan, Endevor, Harvest and ISPW.

VerifIEr: QA & Code Checking

VerifIEr is a static code analysis tool for checking that Gen models are free of errors and compliant with standards. The checks are designed to not only validate the model against standards but also catch coding errors at an early stage in the life-cycle because errors are much easier to fix before time has been spent on generating and testing the code or handing over the code to a separate testing team.

VerifIEr can automate most compliance and QA checking tasks. Automated checking means that a far greater range of checks can be performed when compared with the effort involved in manual checking.

An essential feature of VerifIEr is that QA can be enforced automatically and therefore problems are identified early in the development life-cycle. This contrasts with manual checking or ad-hoc tools where the models are checked at the end of the development phase, by which time the cost of fixing the errors is much higher compared with identifying the issues as soon as they are applied to the model, i.e. before the code has been generated and tested.

pathvIEw: Code Coverage

pathvIEw is a Code Coverage Testing tool for Gen applications, enabling code coverage testing at the action block statement level.

Code Coverage Testing measures the degree to which an application has been tested. By analysing the code coverage results, developers and testers can improve their test cases for functions or statements that have not been adequately tested.

Code Coverage coupled with an intelligent analysis of the results can greatly improve the quality of the application.

Studio Developer: Integrated Development Workbench

Studio Developer is an integrated workbench for Gen development with a modern user interface based on Eclipse. It integrates the IET tools like GuardIEn, VerifIEr and pathvIEw with a completely new and re-designed set of code development tools for Gen. The developer can thus perform their key planning, coding, QA and build processes from within a single integrated toolset.

Tool Integration

Most enterprises will have a variety of tools that support the development life-cycle, from requirements / change / issue tracking tools used for planning and defining changes through source code version control and configuration management tools, testing tools to build / release automation.

Whilst changes can be defined within GuardIEn as Change Requests, often there is a desire for a single point of definition that spans multiple development projects and technologies, so a more generic product might be considered.

At the other end of the cycle, whilst GuardIEn is often used to manage the build and deployment of executable code for Gen applications, where the application is developed with multiple technologies or there is a requirement for production deployment to be managed by a specialist tool, various interfaces and points of integration are available in GuardIEn so that it can manage those steps that are specific to Gen (like object migration, impact analysis and code generation) and then handover to another tool for more generic steps like compilation and deployment, or QA points such as sign-off and peer testing review.

GuardIEn APIs

To enable the integration between GuardIEn and other tools, various high-level APIs are available in GuardIEn.

There are web service APIs that enable the definition of GuardIEn Change Requests and Release Packs for use when another tool is used for change planning and scoping.

There is also a web service API for defining and executing a GuardIEn System Update for use when the Gen aspects of a build/deploy process need to be invoked from an external tool, for example when co-ordinating the promotion of changes within Gen and other development tools.

GuardIEn Interfaces

In addition to the standard GuardIEn interfaces described above, GuardIEn can also be customised to work with other tools using various exits and integration points in the product.

Conclusion

Organisations that are aiming to implement or improve their DevOps processes can include Gen applications within the scope of their DevOps tooling. Generic source code and build automation tools struggle to offer meaningful and reliable automation of the Gen processes due to the complex and unique nature of Gen. However, with the use of IET's tools, these processes can be highly automated to deliver the goals of DevOps, and also integrated into the broader toolset chain in use by the organisation.