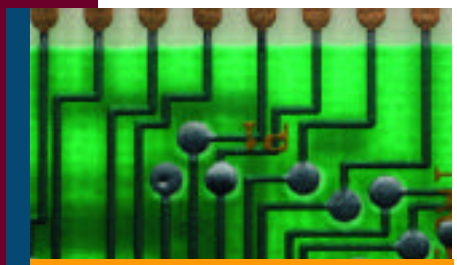


How to Automate Your Application Launch Process Across Environments

BY JEAN-CLAUDE KORTLEVEN

This article examines one way to document an application before it goes into production, generate the application stream (JCL, scheduling, etc.) from this documentation, and control and automate the entire process.

FOR many years, the data center operations staff has faced the lengthy process of launching applications into production. However, this process is not only limited to JCL modification and re-vamping in order to adapt to a target environment. The launch process across environments (test, quality assurance [QA] and production) also includes many other necessary, albeit preventative and error-prone, steps that need to be performed systematically:

- norms and standards must be verified
- the application and JCL must be tested and validated
- the application must be customized to meet the target production environment
- an integrity check must be applied to verify that all application components are available
- documentation must be updated after each maintenance cycle

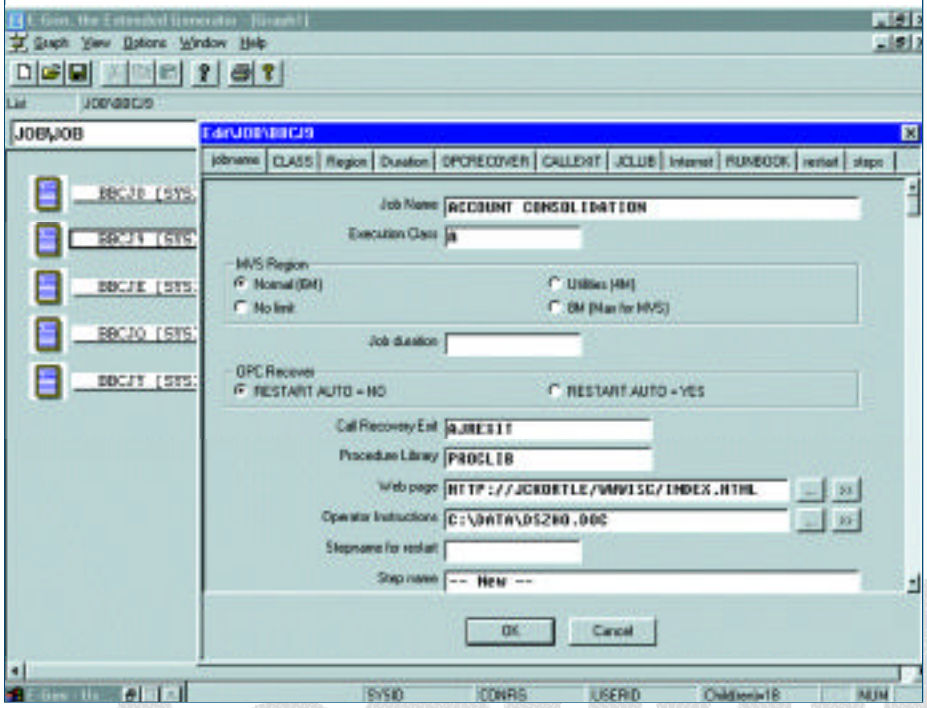
In other words, for each phase of an application life cycle, the staff must perform some major controls. For example, they must:

- check the synchronization between programs, JCL, control cards, load modules, and operator instructions
- normalize the coding for JCL, DSN, DDNAMES and add technical steps
- identify the target environment and adapt specific items, i.e., technical steps, DSN, space requirements
- update the documentation for new programs, new jobs or restart options
- measure the impact of hardware, software or utility changes against current production objects

However, certain recent developments and situations are also causing an increase in the workload. For example, not only the Y2K post conversion validation but also the conversion to the euro, the new single European currency standard that will be launched in January 1999, will require the creation of additional test environments and the coding of the appropriate JCL. Also, in general, applications are getting even more complex; yet, data center personnel are still expected to offer quick solutions, thus leading to a kind of "Quick Delivery syndrome." Indeed, most of the time, data center personnel must launch new types of applications without delay, or they must apply changes on-the-fly without time for test or impact measurement. Finally, multi-center projects have various faces: centralization, site consolidation, even decentralization may enter the picture, thus adding to the concerns of data center personnel.

Until recently, the operations staff has systematically chosen only one tool to solve a specific problem or a combination of products that don't interface with each other. They've tried to automate various tasks using products designed for JCL duplication, verification, alignment and validation. They've also implemented solutions to document the production but realized that once the process was complete, the documentation was already outdated. This is very time consuming and costly when you consider that for every new application or application maintenance, you need to repeat manual operations all over again. This is especially true when you look at the amount of work required to adapt JCL for various environments, define scheduling instructions and make all additional changes. You will no

Figure 1: Job Structure Definition



doubt agree that your documentation is never up-to-date and that you never can perform a correct impact analysis nor rely on accurate operations instructions.

CHANGING TIMES

Wouldn't it be great if you could document an application before it goes into production, generate the application stream (JCL, scheduling, etc.) from this documentation, and control and automate the entire process? Well, it is possible! Let's consider that the documentation becomes the entry point. This means that nothing goes into production before it has been completely and correctly documented. For example, a product such as E-GEN/WS from International Software Company introduces a new methodology for data center automation. This solution is based on an object-oriented repository. The repository resides on the mainframe and contains all the production components. During the implementation phase, you can select which components of your IS legacy must be documented in the repository. Syntactic analyzers will collect the selected production components and populate the repository. Once the IS legacy has been incorporated into the object repository, operations staff can access this documentation using the graphic interface component installed on their PC workstation. From this moment, the repository is the entry point for application creation or maintenance. The operations staff doesn't need to manually code or manipulate the JCL anymore, nor define execution criteria into the scheduling package. The application stream generation will automatically be performed upon request. This is a solution to document, maintain and launch applications into production.

> **"I envision that in the not-so-distant future JCL will be automatically generated from the operations documentation."**
> **George E. Kurtz** [1]

DOES ANY IS LEGACY FIT INTO THIS REPOSITORY?

The central repository can contain all objects used in the day-to-day operations. The operations staff dynamically builds the repository using a set of collectors (syntactic analyzers). They can collect any source of information and automatically build object categories in the documentation repository.

The number and type of categories are dependent on the objects used in operations. For example, the repository can contain categories of objects such as OPC applications, transactions, jobs, various categories of programs (COBOL, DB2, utilities, etc.) and many file categories (Sequential, PDS, VSAM ESDS, VSAM KSDS, reports, input cards).

Each object category has a corresponding template that contains properties and relations. For example, the JCL collector builds templates for jobs, steps, programs or files. It defines properties based on the parameters associated with each JCL statement. The relations between objects are defined to provide easy graphic documentation and navigation. During the implementation phase, an administrator can review and customize each template. The purpose of the customization is to remove common properties,modify them or add new properties in order to define all necessary material for the generation and the graphical documentation.

As illustrated in Figure 1, common object properties such as MSGCLASS, MSGLEVEL, SYSUDUMP and SYSABEND do not need to be present on the JOB documentation template. If it is an operation standard, the generator can automatically insert these properties at generation time. Information such as job name description, execution class, MVS region, restart step, and even OPC restart are JOB properties; therefore, operations staff must be able to update them whenever required. For this reason, the administrator may, for example, want to propose multiple selection options for the REGION and the execution class or provide a larger entry field for the object description. In day-to-day operations this repository customization facility will allow the operations staff to become more specific when creating or maintaining an application. The same customization process can take place when using other collectors to consolidate information coming from other sources such as scheduling information and COBOL source programs. At this stage, the central repository contains all the objects collected from various sources.

In order to guarantee that the documentation is complete and accurate prior to going into production, an administrator can define several rules to make sure that the documentation follows a specific standard. He can check the validity of an object name or length, force a mandatory field and verify

I. George E. Kurtz, You Can't Work in the Dark! *Understanding Automated Data Center Operations*. Published in 1990 by Strategic Technologies Institute, Inc. ISBN 0-9625751-2-7.

Figure 2: View JCL Generation Results

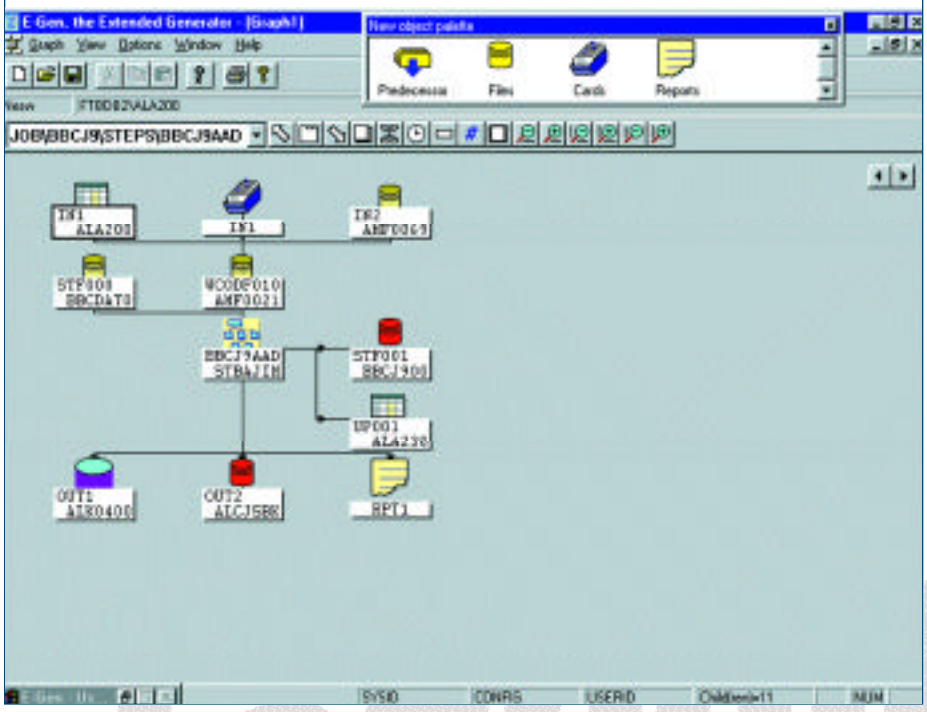
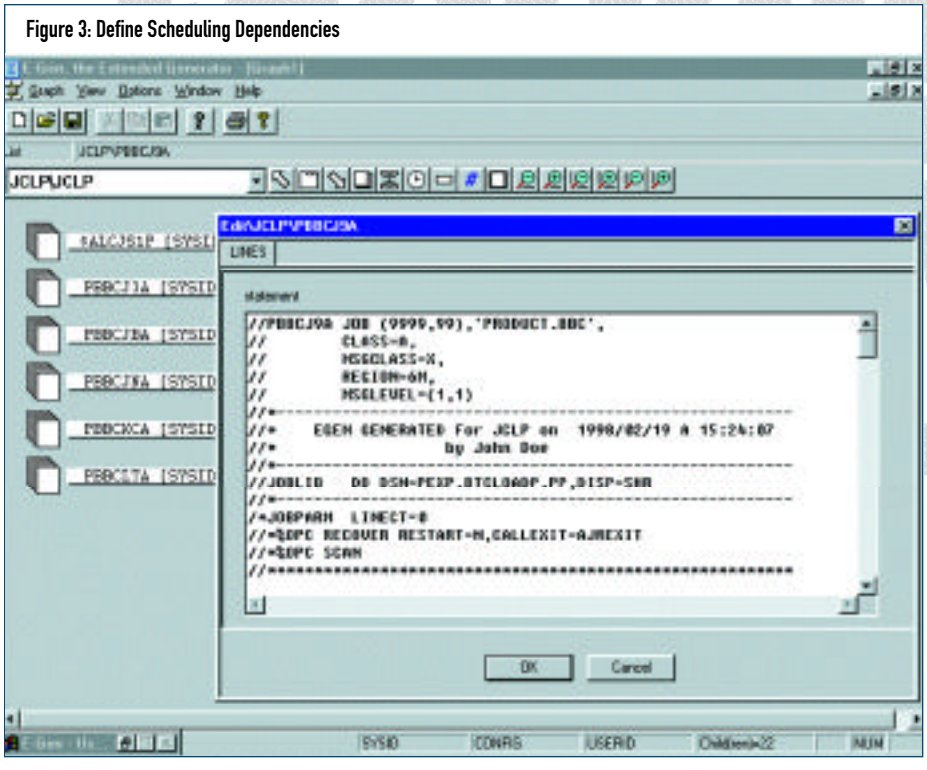


Figure 3: Define Scheduling Dependencies



that object properties are homogenous. At the same time, the administrator can also customize the object palettes. Along with the object templates, the object palettes are permanent assistance tools for the end user. The operations staff has a perfectly secured working environment. They can now centrally use this documentation repository and generate an entire application stream from the documentation.

A CASE STUDY

The following case study examines how E-GEN/WS helped one insurance company to achieve its automation objectives by allowing them to easily maintain their application documentation and define scheduling information. You will also discover how they generated various forms of JCL, inserted the necessary technical steps whenever required and produced a batch loader for their scheduling package, OPC.

First, the operations staff graphically defined the application functional documentation using existing or new objects. They started to perform this task as soon as they received information from the development department. All objects were associated with properties. These properties had default values but the operations staff could change them whenever required. They easily created the graphic documentation using the contextual object palette, a permanent user assistant. As illustrated in Figure 2, the process was as follows:

1. create a job
2. select and order the steps (programs)
3. link a program name to the step and select files icons to associate them with each program

The functional documentation only contained information provided by the development staff. Operations no longer needed to worry about backup and restore, pre-allocation and many other tasks that would take place during execution. These steps, depending on the target execution environment, were automatically included in the JCL during the generation process.

Second, the generation of the various JCL statements, the scheduling definitions or any other required production orders were produced from the operations graphic documentation. When the documentation was ready, they used the "bundle" facility to package the application for generation. The bundle is a snapshot of the application; all the information related to the application (application, jobs, programs and file properties, link between objects and execution criteria) is encapsulated. Once a bundle was created, its content became independent from the global documentation. It represented one version of the application at a given moment. From then on, application modifications could still occur against objects stored in the central repository. They had no impact on the content of the original bundle. When operations decided to bring these new modifications into production, they performed the same packaging steps and obtained a new version of the application.

Finally, the application was available for processing by any defined generator. Each generator was associated with a set of models. Each object of the bundle was, depending

Figure 4: View the Scheduling Batch Loader Results

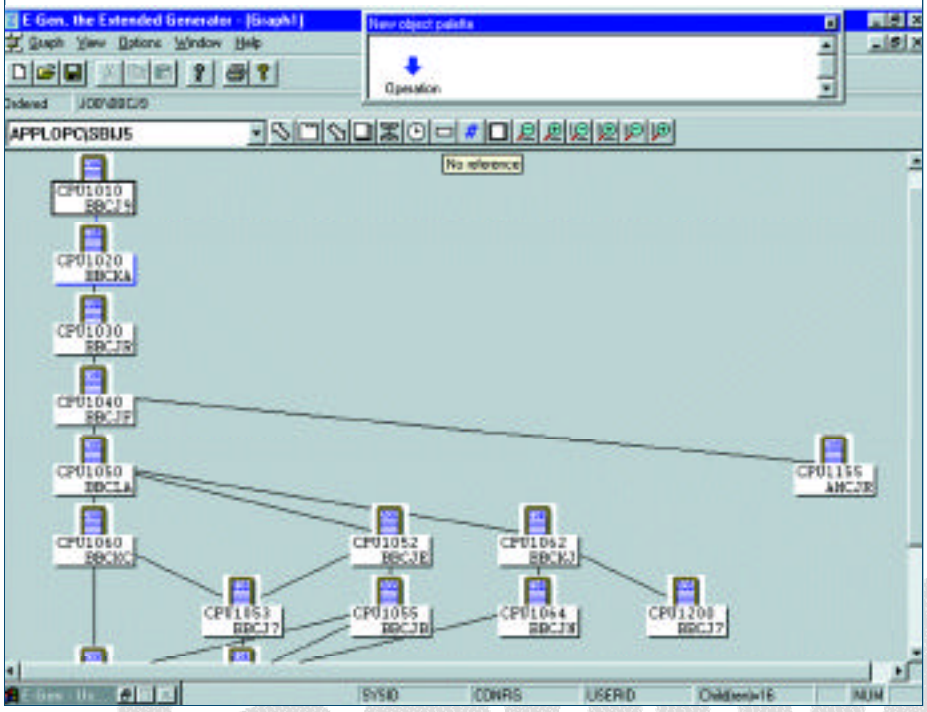
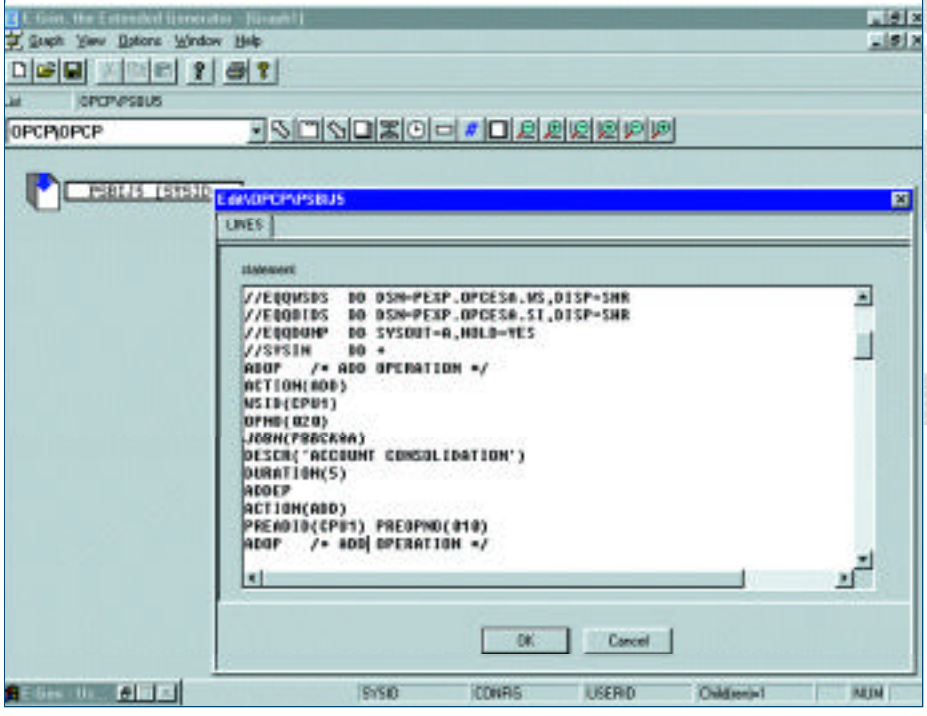


Figure 5: Job Property Definitions



on the generator type, translated into the appropriate operations statement. In the following examples, the generators automatically translated the JOB object into production JCL. In Figure 3, the JCL generator selected the appropriate model and generated the JOBCARD using properties defined in the repository and defaults contained in the generation model. Note that based on the target environment, for example production JCL [JCLP], the JOBNAME was standardized. The generator also inserted a "P" for production in the first position and an "A" as the last character. Based on other job properties, more control cards were also dynamically added: COMMENTS, JOBLIB, even scheduling parameters for OPC [%OPC] and JOBPARM. You might think that the automation level they reached was easy to achieve even though JCL coding and application launch control are a much more complex process to handle. You're right. Let's look further.

When processing JOB objects for a specific target environment, the JCL generators automatically standardized all the statements that are normally error-prone. The STEPNAME along with the COND statements were added, the DSN and the DISP parameters were computed. Along with many other parameters, the JCL generator also computed the SPACE parameters. These parameters are generated using object properties defined in the repository. The generator also used the file size and space allocation factor specific for the production target environment.

In this data center, operations also used production rules such as pre-allocation of all the new files before the start of the application. They chose to perform a full backup before any application began, they executed a selective backup after each update, and they planned a restore step from the latest backup set in case of problems. In this is scenario, the JCL generator automatically inserted all the necessary backup, restore or pre-allocation steps in the JCL. Operations had the guarantee that norms and standards enforcement were strictly applied. Obviously, the generators did not always proceed systematically the same way. Specific norms and standards were applied for test, quality assurance [QA] and production environments.

Once the graphic documentation for the job was ready, operations also defined the execution criteria for the scheduling package. As illustrated in Figure 4, they used the contextual object palette to define the job's execution chronology.

From these scheduling definitions, operations also produced the corresponding OPC batch loader. The generation process was performed separately but could also have been executed along with the JCL generation. In Figure 5, the OPC batch loader generator [OPCP] translated the JOB object and the predecessors using a specific scheduling model, although operations only had to define the JOB properties and predecessors in the documentation repository.

As you can see, launching an application into production using a standard methodology tool became an easy process for operations. Once the documentation was ready, the set of generators translated the object documentation into test, quality assurance and production JCL. The generators also produced

the scheduling definitions. In fact, they can translate any object into any production order as long as corresponding generation models are available. At the same time, the generators took into account the target environment in order to adapt all properties and meet operations norms and standards. Whenever required, they also added the necessary technical steps. In other words, a single manipulation was sufficient to implement changes for all the concerned environments without any error.

In this case study, operations produced various types of JCL and an OPC batch loader. However, it is clear that the methodology they used can be implemented in any data center and will generate production orders for any automation product, i.e., scheduling and report distribution. The operations staff only has to customize the generators. Each generator will react differently to automatically apply the norms and standards to match the target environment requirements.

WHAT IS THE TECHNOLOGY BEHIND THIS?

Operations documentation is maintained in a centralized repository, permitting everyone to work with the same information with online graphic access. This repository resides on the OS/390. The language used to code the engine is C++. This guarantees total portability across platforms. On the mainframe the database uses the VSAM technology with proprietary object organization. The graphic interface is available on Windows 95 and Windows NT. In the case study, the solution uses TCP/IP to support data exchange between the OS/390 server and the various graphic workstations. Each object and object property is associated with its own unique "foot print," allowing operations to perform simultaneous modifications. A built-in logging system prevents object documentation regression and guarantees object integrity. Standard security systems such as RACF, TSS or ACF2 control the various access modes to the repository. Finally, as this methodology is available on PC workstations, the administrator can also choose to activate the "association" property. This facility allows operations staff to dynamically gain access to various pieces of information available on the web or an intranet, in a word processing product, in a spreadsheet, etc. Additionally, this open architecture allows the administrator to plan and perform the necessary data modeling from his workstation, without any interference with the daily production. When this process is complete, he can upload the repository to the mainframe and have it available for operations.

CONCLUSION

Beyond an automation solution you must also always look at the costs, the productivity and the return on investment (ROI). The use of the methodology described in the case study allowed the site to reduce the application maintenance cycle by six, which is the cumulated time required to process JCL and documentation changes across their environments. The same process used to take approximately six hours per application. The application maintenance costs were reduced by nine (taking into account a salary rate of $30 (U.S.) per hour; the maintenance cycle used to cost $180 (U.S.) per application). Finally, based on an average of 200 application changes per year, the overall annual maintenance costs were divided by nine (thus reduced from $36,000 (U.S.) per year to $4,000 (U.S.) per year). Beyond these figures, the direct benefits for day-to-day operations were much more realistic. One new and single working method offered them normalization and standardization, error-free JCL, as well as a true synchronization between jobs, programs and applications. The application launch process across environments has been simplified and totally automated. The application maintenance process cycle has been reduced and secured. Finally, they can rely on up-to-date and real-time documentation containing all necessary information to run the day-to-day operations. As the operations staff says, "We can now face any workload increase along with Year 2000 and euro conversion because we know that we will always be able to deliver the work on time and without any error." ts

Jean-Claude Kortleven is vice president of sales and marketing for International Software Company. He has more than 15 years experience in data processing. His experience includes operations and system support for a large service bureau as well as working as a systems engineer and pre-sales engineer for a leading software vendor. He can be contacted at jckortleven@iscsoftware.com