THE VISUAL FACTORY SUITE: FACING EVOLVING MASS PRODUCTION IN SPATIAL DATA PROCESSING ENVIRONMENTS

Dr. José Navarro
Institut Cartogràfic de Catalunya, Spain
pep@icc.es

KEY WORDS: Automation, Semi-automation, Integration, Data processing, Industrial, Software, Production.

ABSTRACT

The construction of software systems assembling multiple components and devoted to massive batch production ("factories" in this context) has always been a slow, difficult task, delivering rather inflexible solutions, requiring specialized personnel (software engineers) not only to create such systems but, often, to exploit them. The reasons for such difficulties may be grouped as follows:

- Assembly. Combining several programs to create a workflow is a task posing technical problems.
- Change. New products are requested; improved algorithms appear; new data formats have to be used, so new or modified systems are necessary.
- Complexity. Domain experts are necessary to determine "how things have to be done". Usually, software engineers are not aware of the in-depths of the discipline they are building a system for.

Such difficulties lead to stiff software systems, far from being able to face the challenge of every-day production. The **Visual Factory Suite** (VFS) for Windows NT is a set of applications and standards created at the Institut Cartogràfic de Catalunya (ICC) devised to alleviate or solve these problems.
The main objective of the VFS may be stated as follows:

> *The production (not the software) engineer must be able to create new factories by him(her)self and put them into production in a few minutes.*

This objective has been fulfilled with VFS 1.0. Using the VFS, the production engineer may:

- Design factories in a *visual* way.
- Deploy these factories to perform actual production according to the available resources and,
- visually again, monitor and control the production process.

1 INTRODUCTION

Flexible mass production is an objective that has been pursued by the ICC since so long ago as 1987. By that time, the production of a 6.000 sheet 1:5.000 B&W orthophoto series (among others) had shown that no manual orthophoto system would be able to give a throughput high enough as to guarantee the fulfillment of deadlines. As an answer to this challenge, a brand new, automated batch system was developed and has been in constant operation since 1988.

ORTO (Colomina and Navarro, 1989, Colomina and Navarro, 1991, Colomina et al., 1991, Colomer, 1987) the system mentioned above was devised with some important design aspects in mind. One of the most important was, perhaps, *flexibility*.

It cannot be said that ORTO has not been a successful investment. In many aspects, this system has even exceeded the initial expectations that were conceived when it was first put into production. For instance, and thanks to its pipelined architecture, it is able to deliver up to 140 B&W orthophoto mosaics (6000×4000 pixel, Alpha Server 5/466, OpenVMS) per day with no human intervention at all. However, and in spite of all its advantages, ORTO still lacks flexibility. The system had to be able to accept an unlimited number of "production schemes" or ways to create a product. These schemes, workflows or *factories* from now on, were defined externally, so creating a new product should be as easy as writing an ASCII file containing a new factory.

Typically, a new factory requires between two and three man-weeks to be added to this system. New user interfaces able to cope with the requirements of the new product as well as the assembly of the programs used to perform the actual

production are examples of the tasks that has to be done to get the system working. And the worse of it all is that these tasks must be done by a *software engineer*, that is, someone who is not involved with the production process and must be an expert in ORTO.

Perhaps this is the most important problem. Since the inclusion of new factories is responsibility of a software engineer and the response time is high, the production engineer finds very difficult, if not impossible, to test new ways to produce. This not only concerns new products, but also the maintenance of existing ones.

In April 1998, the ICC decided to improve ORTO. One of the main objectives was to use a much more common platform: Windows NT running on Intel processors. Since the project was scheduled as mid-term and ORTO would be working in parallel—so the actual production tasks were not compromised—a major improvement strategy was devised. The Visual Factory Suite project was born.

2 OBJECTIVES

The main objective of the VFS project was to develop a system able to:

- Provide the production engineer with the ability to design factories in a *visual, foolproof way*, where factory stands for "the way a product is created." Visual design helps to reduce—never to eliminate—the **complexity** of creating or maintaining factories.
- Create the mechanisms necessary to put these factories in a real production environment in just a few *minutes, with no intervention on the side of the development team*, thus reducing dramatically the "time to market". This means that no development tasks must exist at all between design and exploitation: just "design and go".

  Such mechanisms *fully remove* the **assembly** problem. Moreover, combined with visual design produces a highly effective mechanism to accomodate **change**. The only thing to do is [re-]design the factory and put it into automatic production immediately.
- Give the appropriate, visual tools required to monitor and control the actual exploitation of the system, again in the hands of the production team.

It is important to remark the "end-user-philia" shown by these objectives. VFS was born with a clear vocation: leave as much production tasks to those that have the know-how and let the developer's team interfere only when they are really required (the creation or maintenance of "Value Packs", see later on). A complete set of hypertextual, browser-enabled user guides has been provided to boost this end-user strong inclination (Navarro and Ríos, 1999–2000).

Note that although the Visual Factory Suite project may be considered as the heir of the digital orthophoto production system currently running at the ICC, this suite was conceived as a general tool. Therefore, two different lines of work were defined: the VFS itself and the "Orthophoto Value Pack" (OVP) sub-projects.

A "Value Pack" is nothing but a set of programs—adhering to some standardization guidelines—providing with a group of functionalities for a given subdiscipline that may be integrated into VFS . For instance, the OVP includes some typical orthophoto-related programs: digital image rectification, image enhancing, mosaicking and so on. The main function of this pack is to provide the Visual Factory Suite with orthophoto production capabilities.

In spite of the effort devoted to the migration of the programs included in the OVP, it represents a minor part of the complexity inherent to the project. In general, it may be said that these programs are almost identical to the versions that were used in the former environment. Slight modifications have been made to adapt this software to the VFS idiosyncrasy. Therefore, almost no further mention neither of the OVP nor of orthophoto production will be made later on in this paper and only VFS will be described here.

2.1 Technical commitments

Once the objectives described in Section 2 were fixed, a technical plan to fulfill them was developed, which, basically, relied on the following features:

General system. A suite as VFS should be related to no specific discipline at all. On the contrary, the suite had to be able to design and exploit factories creating products of *any kind*.

But, of course, discipline-related software is necessary, since it is not the same to compute GPS bases than rectifying orthophotos. Therefore, the specific programs doing these tasks had to be grouped in separate "value packs" (or plug-ins or add-ons). The OVP mentioned above is the first example. Note that the VFS is good for nothing if no value packs are included.

Standardized "Value Packs" architecture. Perhaps one of the greatest technical challenges faced by VFS—and where ORTO, if not exactly failed, could have been improved a lot—was to assemble *automatically* a set of programs running in a predefined sequence to create a given product—that is, running a factory.

To make possible "on-the-fly" assembly, that is, make the "design and go" objective a reality, the programs in a value pack must adhere to a set of standardized rules, covering both aspects of semantics characterization—what does the program do?—and programming techniques—how it must be done?

Accessible to all levels of developers. To really take advantage of the VFS' capabilities, value packs must be developed, covering whatever discipline of interest. The technical requirements for such a task had to be simple enough as to permit any non-specialized programmer to do it. This was another of the main problems with ORTO: a very specialized software engineer, with a deep knowledge of this system's architecture was necessary to add software implementing new functionality.

Human intervention integration. Full batch production is a dream, since not every process may be completely automated. Often, algorithms solving a problem in an automatic way do not exist and it is necessary to use the operator's know-how to perform a task manually.

This kind of situation leads to interrupts in the process of the automatic—batch—parts of the factory and, if not handled correctly, increases the complexity of the management. Therefore, VFS had to be able to include human interventions during the *design*—of factories—process and handle them in an integrated way. This means that the suite will be able to create *complete* products, not only components.

Scalable and distributed. The VFS had to be able to run in as many computers as necessary to increase throughput. On the contrary, it should be able to run in a single computer if requested.

Robustness. Mass production systems constantly have to face lots of real-life problems: software misbehaviour, power supply failures, disk space exhaustion, network collapses or operator's misuse are some examples of these undesirable situations. Any real industrial system has to be able to cope with them gracefully, avoiding as much as possible the loss of invaluable data and hours of work. Even in the case of a major equipment or software breakdown, they must be able to restart with no human intervention at all. VFS was designed to meet these quality standards.

Open to Production Management Systems (PMS). VFS is, basically, a pure production system taking no care about management issues. Since mass production requires a great deal of management tasks, the seamless integration with PMSs became a must.

3 THE VISUAL FACTORY SUITE AT A GLANCE: THE PRODUCTION LIFE CYCLE

The VFS is not only a set of tools but also an idiosyncrasy, a model defining how production is driven. This model, or "production life cycle" could be summarized as follows:

1. **Design**. The production engineer designs or maintains factories, using the visual tool provided for this purpose. When designing a factory, the engineer may combine *only* the programs provided with the value packs installed along with the VFS itself and may include as many human interventions as desired. Once a satisfactory result is achieved, the new design is stored in a database.

   Note that the design thus produced is an "abstract" one and *no assumptions are made about physical resources' allocation* (computers, directories, disk quotas, etc.) These aspects are solved in the following phase.

   Figure 1(a) shows the aspect of the Visual Factory Builder, the visual tool used to design factories. In this figure all the basic building blocks used to "draw" a factory may be seen: programs (big boxes with little boxes representing files on top and bottom sides), human interactions (the same as programs, but with a red box on the right side), repositories ("places"—**not actual directories**) where information is stored, and connections (lines between the previous elements) showing how information (files) flows.

   Special mention deserves the fact that designing factories in this way leads to *self-documenting workflows*: to make production possible, the workflow must be "drawn" *a priori*. Moreover, this documentation will always be up to date, since modifications in a workflow must go through a change in the original design.

2. **Deployment**. This phase is the keystone to organize the production process. Basically, the production engineer takes the necessary actions to distribute the work among several projects, selecting and customizing factories and assigning actual physical resources to do the process.

   The deployment activities are based in a hierarchical project / factory / computer paradigm:

- *Projects*. Are used to organize the production process. The production engineer may define as many projects as desired.
- *Factories*. A project may use an unlimited number of factories, that is, different ways to produce. Note that a factory may be used by several projects simultaneously.

  The important point here is that each project / factory couple may be customized to optimize the production process. For instance, it is possible to preset the values of some parameters (used by the programs in the factory) so they will be used for all the items produced and will never be asked to the operator—saving operation time. Therefore, the *same* "abstract" factory created in the design phase is used in slightly *different, optimized* ways depending on the project they are attached to.
- *Computers*. Also known as "production lines" in the VFS jargon. Each project / factory couple may be executed in as many computers as desired. In this way, there is no limit to the throughput that may be obtained. Again, a computer may execute several project / factory combinations, boasting flexibility.

  Figure 1(b) shows the main screen of Factory Planner, the tool used to deploy the production. Note the hierarchical organization of projects, factories (attached to the selected project) and the names of the available computers (to run the selected factory).

  Computers do the real work so they need to be aware of every physical detail concerning the exploitation. This means that for each project / factory / computer triplet introduced, the following information must be provided with: directories to use and disk quotas to keep.

  Figure 1(c) represents the actual screen used to select directories and fix quotas for a given computer.

The minimal setup required to work is, therefore, a single triplet, defining a project containing a factory to be executed in one computer.

The typical *whole* deployment process takes between 10 and 20 minutes, depending on the complexity of the customization made to the project / factory couple, and immediately after, the VFS *is ready to produce*.

3. **Exploitation**. Once a factory has been designed and deployed, the production process may be initiated. To make it possible, two tools are necessary.

   - *Launcher*. The operator needs a tool in order to initiate the process of new items; it must provide with the mechanisms necessary to specify all the parameters related to the production of the item, as the project / factory / computer triplet (which identifies how and where the process will take place) and any other relevant information required by the programs integrating the selected factory.

     After all the necessary information has been introduced, the launcher gives the appropriate orders to initiate the process of the new item.

     Figure 1(d) presents the Factory Launcher tool's main screen, where, again, projects factories and computers are shown, along with the item-specific information necessary to process it.

     It is important to remark that Factory Launcher is the *sole* tool required to launch production. The normalization process undergone by the value packs permits an absolute general launching system, not affected by the different programs included in different factories. Therefore, *no user-interface customization is needed*.
   - *Dispatcher*. Factories must be executed, that is, the programs they are composed of must be run in the correct order, using the right files and in the selected computer. This is the task assigned to the VFS' dispatcher.

     Factory Manager is the tool taking care of these tasks. When running, it loads the definition of the available factories as well as the deployment specifications; then waits for requests to produce (sent with Factory Launcher) and runs the programs in the appropriate sequence.

     This tool also takes care of tracking the possible problems detected during the production problems (missing executables, network problems, exceeded disk quotas and son on). Whenever an anomalous situation is detected, Factory Manager notifies the monitoring tool (see later on) about the problem, so the (human) production manager may initiate the appropriate corrective actions.

     Finally, Factory Manager keeps a log database containing information about the production process (which also records the items that could not be completed—and the reason of the failure).

     Each computer devoted to production must have a copy of Factory Manager installed and running.

4. **Monitoring and control**. Once production is going on, it must be tracked. This means not only that information about the precise state of the system must be provided as soon as requested but also that mechanisms to control *how* production is performed must be included.

   Moreover, and since human interaction is fully integrated in the VFS exploitation idiosyncrasy, the availability of tools able to handle this type of actions is a must.

- *Basic monitoring and control*. Visual Factory Tracker is the tool taking care of these problems. First, it is able to show, in a graphical way, the state of all the items being processed by VFS (see Figure 2(a)). This includes details about projects, factories, computers, timing, progress and even the precise program (within the factory) that it is being executed for every item in the system.

  Additionally, it is possible to interact with the production system. Examples of interactions are: priority control (to modify the order in which items are processed), item deletion, graceful shutdown (to stop the VFS in case of emergency, as power supply failure and UPS backup) among others.

  Finally, the Tracker behaves also as an alarm reception center. As mentioned above, the dispatchers take notice of every anomalous circumstance detected during the exploitation process. Visual Factory Tracker collects all these alarms and notifies the operators in a centralized way, so appropriate measures may be taken to correct the problems.
- *Human interaction*. The tool to use in this case is Visual Factory Interacter. Interacter is, basically, a tracking tool that takes notice of all the human interaction requests sent by the dispatcher(s) and, afterwards, shows them on screen. Enough information is provided with the request as to allow operators to perform the solicited (manual) task. Once this is done, Interacter is able to notify the dispatcher about the availability of new data, so the later tool may continue, automatically, the process of the item affected by the human intervention.

  Figure 2(b) illustrates Visual Factory Interacter in action. Two request are shown; the first one is a expanded view while the second one is represented in a collapsed form (both representation methods are operator-switchable). In the first case, all the details describing the human intervention (type of task, input and output files, etc.) and clearly visible.
- *Log viewer*. Last, but not least, Factory Log Viewer is a tool devised to browse the log database maintained by the dispatchers. Using the viewer, it is possible to check whether an item was correctly processed or not.

4 INTEGRATION WITH PRODUCTION MANAGEMENT SYSTEMS

The Visual Factory Suite is focused in providing with a solution for a problem that could be stated using a few words: *how to produce efficiently*. But mass production is affected by other problems besides the ones that VFS solves or alleviates.

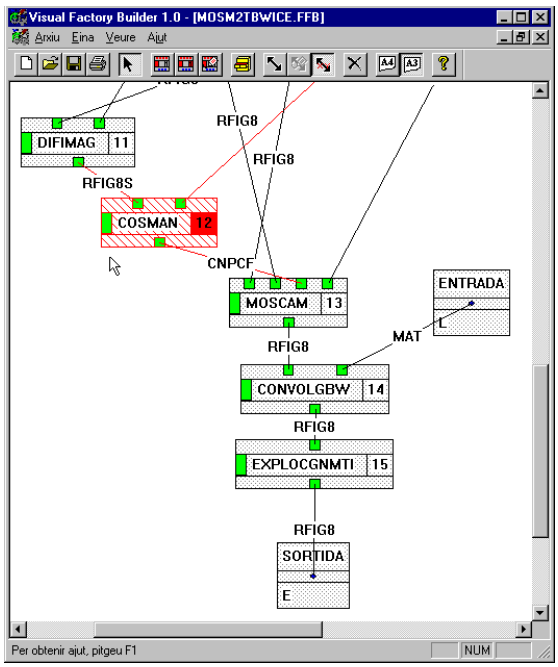
When dealing with a low number of items, the management tasks related to the production process could be considered of little of no importance, for the size of the problem is small and may be handled reasonably well with little or no management tools at all. In the case of true mass production, the situation is the opposite, and the availability of a *Production Management System* (PMS) is a must.

Obviously, a PMS should take care of all the tasks related to the management of the production process. Examples of such tasks, among many others, could be: collect, store, organize the information necessary to process an item and help the operators to determine the items that are ready to be processed since all the necessary information has been collected; also, it should be able to minimize the interaction with a pure production system (as VFS), using tailored end-used interfaces, to reduce operation times, as, for instance, creating packages of items to be processed with identical options. Not less important, a PMS should also provide with a broad range of status-tracking utilities able to give a clear representation of the state of the system, including information about the production problems that always arise when dealing with mass production environments.

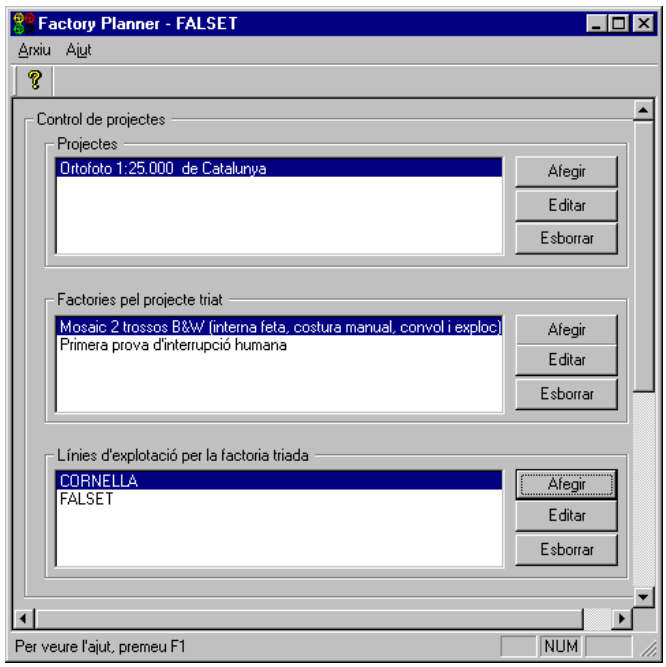
Building a PMS is a discipline-related task, since different types of products require also different management techniques and utilities. Notwithstanding this, the VFS offers a standardized way to integrate its pure production services into a PMS: the FactoryLink technology. A PMS using FactoryLink may be organized in the way most appropriate to manage the production process it is targeted at, so no restriction about its design or implementation exist. However, it may use all the services included in VFS to *initiate* the actual process of items, providing that all the information necessary to do it (project / factory / computer triplet as well as the parameters used by the programs included in the factory) is conveniently supplied.

As described in Section 3, Factory Launcher is the default tool used to start the process of new items. In fact, it should be used *only when a PMS is not available*. Having a FactoryLink-enabled PMS has important implications related to ergonomics and throughput. Perhaps the most important are that (1) the interface used by the operator to launch the process of new items may be fully customized to the needs of the product the PMS is targeted at and (2) many, if not all the parameters necessary to initiate the process may (or should) be stored in the PMS databases, thus avoiding the need to provide them manually using the keyboard. (Ideally, a good PMS using FactoryLink should never ask any data to its operators when launching an item, since all the information should be available on line).

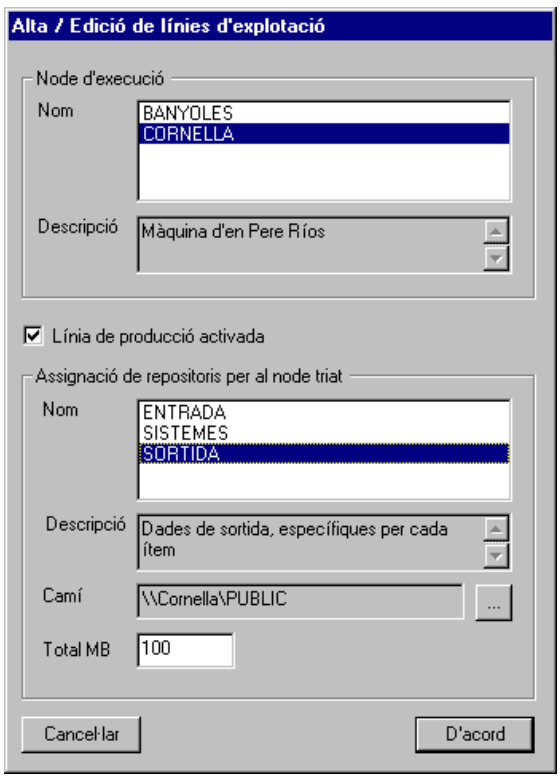
Since the first project using VFS technology at the ICC deals with digital orthophoto production, a PMS specifically designed for this type of product has been successfully developed using FactoryLink technology. This product is installed



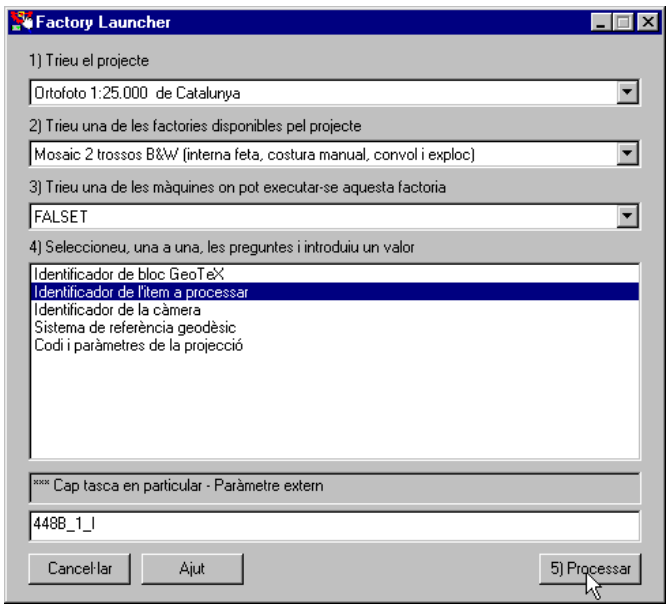
(a) Visual Factory Builder



(b) Factory Planner's main screen

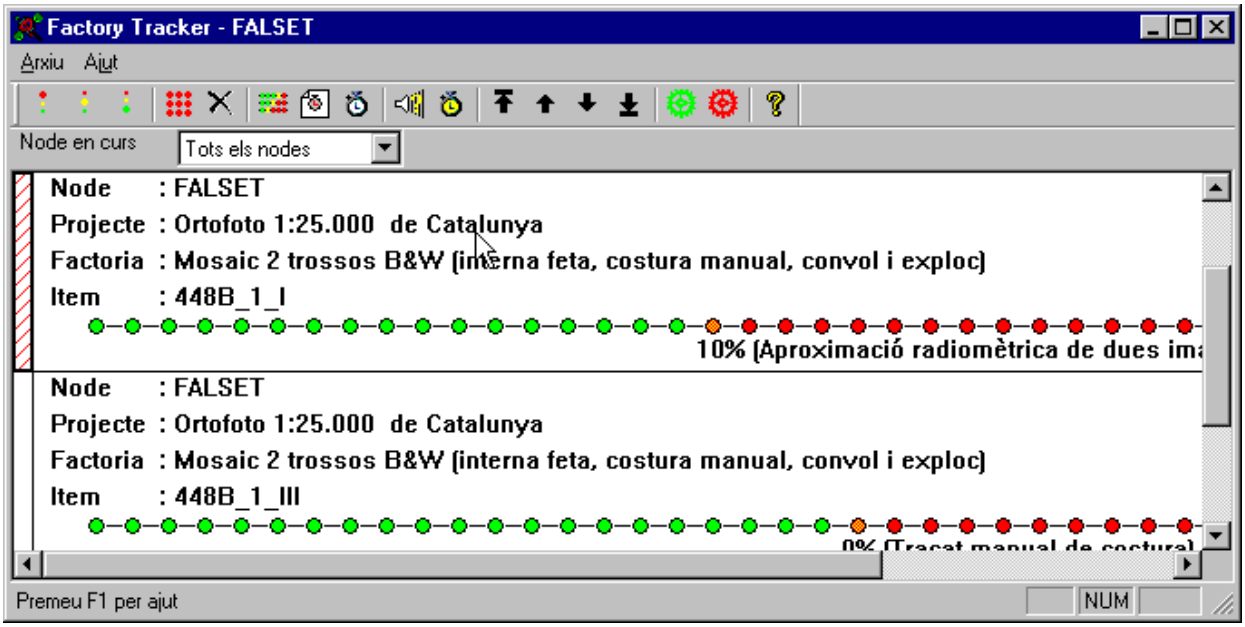


(c)Assigning resources in Factory Planner

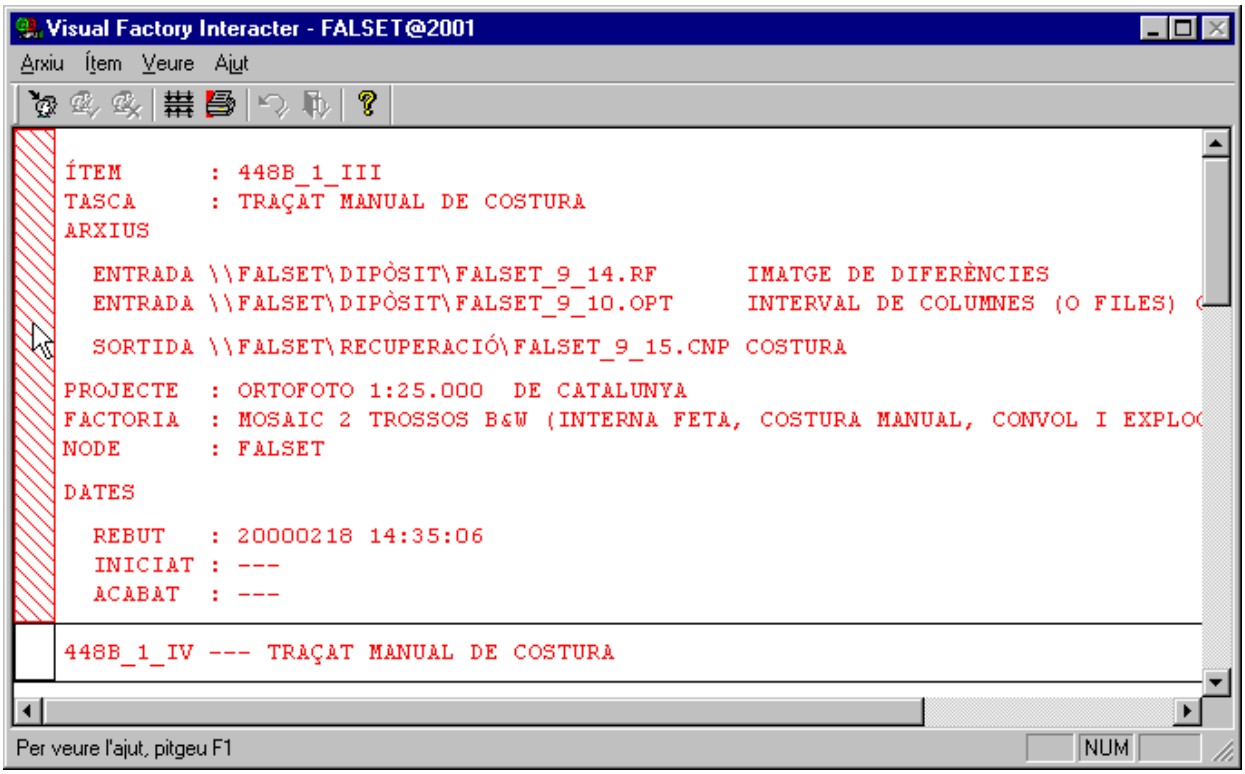


(d) Factory Launcher

Figure 1: Actual screen shots of some of the VFS tools (1)



(a) Visual Factory Tracker



(b) Visual Factory Interacter

Figure 2: Actual screen shots of some of the VFS tools (2)

along with the Orthophoto Value Pack for VFS mentioned in Section 2, and served as a field-test of the concept. Of course, Factory Launcher will never be used in this specific environment: the tool provided by the PMS, OrthoLauncher, is much better.

5 CONCLUSION

The Visual Factory Suite is a set of tools very well suited to face the challenges found in mass production environments. Its visual design tool, combined with the automatic exploitation facilities it provides, help to make the classic problems usually found in this type of systems—complexity, assembly and change—almost disappear and leading to very low time to market ratings; additionally, the know-how of the production team is directly invested in improving the production system, with no intervention on the side of developer teams, whose only responsibility is to create specific value packs.

Born as the heir of ORTO, a seasoned industrial system with more that 10 years of steady work on its back, the VFS is also a robust system, able to make front to the demands of every-day production. Furthermore, it is scalable and distributed, and may be exploited either in a single personal computer or in a large network.

The VFS is related to no specific fields of application; on the contrary, it is a *general* set of tools. This means that it may be used whenever a mass production schema is applicable. The "Value Pack" add-on model is the way to bring genericity down to specific fields of application. The Orthophoto Value Pack for VFS is the first real example.

Value Packs must be seen under two different light sources: first, the discipline they cover; of course, the algorithms used in a value pack devoted to digital orthophoto production will never be the same as those included in a GPS pack. But the important point here is the second standpoint: how to make these packs VFS-compliant. A set of very simple design and programming standards make this task accessible to software engineers with no previous experience in the Visual Factory Suite realm.

Although it was conceived as a closed system, taking care of pure production tasks, the VFS provides with the mechanisms necessary to facilitate the integration with production management systems via the FactoryLink technology. An example of such integration is the OrthoLauncher tool included in the orthophoto PMS developed at the ICC.

After a period of several months of "laboratory" tests, the Visual Factory Suite was put into "real" production on March 1st, 2000. The first project, again, was a test one. The purpose, checking the system in an actual industrial environment. By the time this paper is published, the VFS is expected to be working under normal conditions in a steady way.

REFERENCES

Colomer, J. L., 1987. Configuring a production system for digital orthophoto generation. Technical report, Institut Cartogràfic de Catalunya, Barcelona, Spain.

Colomina, I. and Navarro, J., 1989. El sistema de generación de ortofotos digitales del ICC. In: Sistemas Cartogràficos Digitales Españoles: productos y desarrollos. Primera Jornada Técnica de la SECFT, Madrid, Spain.

Colomina, I. and Navarro, J., 1991. On functional requirements of a photogrammetric station for digital orthophoto generation. In: Sistemas Fotogramètricos Analíticos Digitales: una nueva generación emergente. Tercera Jornada Técnica de la SECFT, Barcelona, Spain.

Colomina, I., Navarro, J. and Torre, M., 1991. Digital photogrammetric systems at the ICC. In: H. Ebner and C. Heipke (eds), Digital Photogrammetric Systems, Herbert Wichmann Verlag GmbH.

Navarro, J. and Ríos, P., 1999–2000. Documentació de la Visual Factory Suite (set of hypertextual documentation about the Visual Factory Suite, in catalan). Barcelona, Spain. ICC's inner documentation.