Planning By Priority

*How to prioritize project tasks to maximize delivered business benefits*

*A whitepaper by*

Andy Carmichael, PhD, CEng, FBCS
*Vice President International Operations, Ivis Technologies*

**Issue 2.0**

Ivis Technologies
5110 N. 44th St
Suite 200L
Phoenix, Arizona 85018
United States
Tel: +1 (602) 343-8200
Fax: +1 (602) 926-2348

[www.ivis.com](http://www.ivis.com)
Planning By Priority

How to prioritize tasks to maximize delivered business benefits

Abstract

In today's highly competitive business environment, it is absolutely critical to focus on what is important, versus what is "next on the list" – in other words to plan by priority. Priority-driven processes do exactly that and give you far more accurate forecasting capabilities than traditional project planning approaches.

Projects which plan by priority use the importance and cost of key deliverables to dynamically drive the schedule rather than following predefined activities. So your focus is on what provides the greatest payback soonest. This makes the organization far more responsive to mission-critical opportunities and challenges. Agile software development processes, as well as many other business processes, use such an approach in preference to "waterfall" style project lifecycles. The priority of an activity is the primary driver deciding task order (rather than ordering different types of activity in different phases of the lifecycle). This focus on priorities can lower risk, increase efficiency and optimize resource utilization - resulting in faster deliveries of key requirements.

This paper, the first part of a two-part discussion of priority-driven processes, looks at applying the process improvement environment xProcess to a priority-driven project, and it explores the “how-to’s” in setting up and scheduling a small project. The second part [1] considers how to design processes that use prioritization alongside the project patterns, task patterns, artifact templates and tailored quality-checking procedures that the environment supports.

The paper continues the series on Dynamic Process Management (DPM) and xProcess, looking at the ability of these techniques to enhance agility and transparency within organizations.

Introduction

Priority-driven processes work by focusing on the deliverables of projects and asking the stakeholders to say which of the many things that the project could deliver are the most important. The planning of such projects proceeds in a different manner to that of “waterfall” style approaches since each deliverable has, as far as possible, an independent lifecycle and dependencies between requirements are kept to a minimum. This gives stakeholders the ability to change the priorities of requirements much later in the project without disrupting the work, and benefits the business by providing more agility and responsiveness to changing circumstances.

Agile software development processes are just one example of the types of processes that use a priority-driven or feature-centric approach [2]. Methods like eXtreme Program-
ming, Feature-driven development, Scrum, Evo and Crystal all ensure that the specification of requirements (variously referred to as user stories, features, backlog, use cases, and so on) is carried out individually, so that the lifecycle of the specification and fulfillment of each requirement is as far as possible independent of other requirements. As a result stakeholders can prioritize (and re-prioritize) requirements. Prioritization becomes one of key project controls for managers, alongside other primary controls such as changing the project scope, resource levels and release dates.

Software processes however are not alone in benefiting from priority-based controls. Activities such as marketing, legal services, estate management, mergers and acquisitions, scientific research, professional sports, entertainment and manufacturing can all apply similar techniques. If the goals of the projects can be enumerated and if, as is usual, it results in a list that is longer than initial budgets or timescales might allow, a priority-driven approach is called for. The process behind each goal or requirement should be considered separately and then the scheduling of the whole project can proceed in priority order.

In this paper we will start with the consideration of a simple project defined within xProcess in terms of just four process patterns: a project, a task, an overhead task and a timebox. We show how projects can be built and scheduled with just these components. We then demonstrate how these and other process patterns might be enhanced and defined to support projects following more specific processes. The “artifacts” of such processes – the documents, products and other files produced by the project – can be built from templates, linked to other tasks and artifacts in the project, and managed by xProcess’s versioning and control system. This focus on process definition is developed further in the next paper in the series.

**A simple project**

The planning environment provides a base level process (called the Root or Simple Process) for creating projects with a minimum amount of predefined process. It’s a good place to start since it is easier to work from too little process to the ideal amount than from too much. When you create a new data source (the repository that can be shared with other users), it will already contain the Simple Process. While you can change this process (or more usually create new processes which extend it) you can in fact start defining projects straight away without defining any additional elements of process.

Figure 1 shows a user instantiating a “Simple Project” which is the name of one of the process patterns defined in Simple Process. Having done this there are two more simple steps required before a project schedule can be created: adding the tasks and adding resources.

This example project is a team activity to develop a retail web site for a small company’s products involving up to 4 people and aiming to deliver the first version of the site within a few weeks. Here is the initial list of tasks that the team has arrived at from their first brainstorming session:
- Install web server
- Design home page
- Landing page for product A
- Landing page for product B
- Landing page for product C
- Landing page for product D
- Add shopping cart facility
- Add registration page

- Link to payment web service
- Pre-register current users
- Link in search facility
- Optimize for search engines
- Acceptance tests for complete site
- Support feedback page for users comments
- Add animated demo of product

This set of tasks can be input in a comma-separated list (maybe pasted from a document like this one). We also need to add the people in the organization and link those individuals we want to work on this activity with our project (we can use drag-and-drop for this). At this stage xProcess has enough information to do an initial schedule for the project, using assumptions about task size from the Root Process, and to display a Gantt chart of the plan.

Figure 1. Creating a simple project

In practice we are likely to want to change the estimates for tasks using the task editor. Figure 2 shows a task editor for the task “Add shopping cart facility”. We can see the estimate for this task is being changed to 40 hours (best case 20 hours, worst case 60 hours) and at the bottom of the editor that the number of people expected to work on this task has been increased from one to two.
Figure 2. Changing three-point estimates for tasks

Three-point estimates like this enhance the amount of information that can be incorporated into the forecasts for projects. We can obtain not only a predicted end-date – this is the 50% forecast which is just as likely to be late as early – we can also obtain more conservative forecasts which have 75% and 95% probabilities respectively of being complete on the given date. As we will see, the 75% and 95% end dates are shown on Gantt charts of the tasks as extensions to the main task bars (see figure 4 for example).

Prioritizing and scheduling

With traditional planning tools a lot of emphasis is placed on defining dependencies between tasks so that they will be carried out in the correct order. While this is still possible in xProcess, if we wish to apply priority-driven approaches it is better to leave tasks as unconstrained as possible, and use the priorities as the main way to order tasks. Schedules can then be modified automatically when resources, priorities or other factors change.

In some cases it is genuinely impossible to carry out a task unless the prior task has been completed. In these cases we use constraints on the task to state that either its start or end cannot occur before a given date or event. For example figure 3 shows a user adding a constraint to the “Pre-register current users” task. It is not possible to do this task until
there is a registration page facility in the system so the constraint is added to inform the scheduler of this dependency.

Constraints may reference the forecast dates of other tasks (in this or other projects) and they may also use calendar dates or other dates entered by users such as the target dates on the completion of work packages. An offset may also be defined – by default it is one day, causing a task to be scheduled on the day after completion on the previous task – which means that anticipated delays, such as the typical wait time on external bureaucracies, can be anticipated and planned.

Figure 4 shows the Gantt chart resulting from our work so far. At this stage only one constraint has been added (that the “Pre-register current users” task must start after the forecast end of “Add registration page”) and so the other tasks will be scheduled as soon as possible but in an arbitrary order. This is therefore the point to introduce some priorities.

The task “Add animated demo of products” is clearly a time-consuming activity. While it may enhance the overall system the customer may believe it is not worth delaying the launch of the site for. Features like “Add shopping cart facility” and “Link to payment web service” would no doubt be prioritized higher. Similarly “Support feedback page for users comments” might not be considered essential in the first iteration and so could start later in the schedule.

There are many different ways to approach prioritizing tasks. For example

- assigning tasks to timeboxes and prioritizing them within each timebox
- categorizing tasks (High, Medium, Low or, if you prefer the “MoSCoW” acronym, Must-have, Should-have, Could-have, and Won’t-have) or
- voting for features (a prioritized folder of tasks given to each voter).

The simplest method with only a small number of tasks, and therefore the one which is appropriate in this case, is to have just one prioritized list for the whole project and then we can place the tasks in order. Figure 5 shows just such a prioritized list with all the tasks from this small project. The user is dragging the “Install web server” task in order to “move above Design home page”. Repositioning tasks in this way enables a view of the critical tasks to evolve. Some of the tasks have equal priority (such as the “Landing
“page” tasks for the various products) in which case the scheduler is free to order these in any way.

Figure 4. Initial Gantt chart for the project

Figure 5. Modifying the priority order of tasks
As tasks are re-ordered the scheduler automatically re-adjusts the assignment of team members to tasks – always respecting any assignments that have been made manually, say by the project manager. It also takes into account those “active” tasks, which participants have indicated are currently being worked on, and the required role types for the tasks which must match those of the assigned team member. Team members are selected to ensure the highest priority tasks are completed as soon as possible.

The resulting schedule from this re-prioritization is shown in the Gantt chart, Figure 6. Some tasks now complete later but the highest priority tasks complete sooner in the schedule allowing the team to consider releasing part of the system earlier.

Prioritized plans are amenable to this kind of thinking. If most of the benefit of the system can be derived from just a proportion of the total development effort, making such an early release is likely have many benefits for the users and the development team. Early deployment could result in earlier revenues from the system and feedback from the initial use of the system can result in improvements being incorporated in the final system. Often some of the lower priority requirements are superseded by new requirements which are derived directly from early use of the system.

Figure 6. New schedule of the prioritized tasks

At this point it makes sense to think about target dates for the system completion based on the task estimates and project resources. This is being done in Figure 7 which shows both setting the target end date for a set of tasks and the resulting status of the tasks. Target dates can be set from current forecasts, if you are satisfied that the currently predicted schedule is suitable. This allows each task to be monitored for progress against when it was expected to finish when the targets were set. Negative changes to resources, estimates or simply additional tasks will show up as status changes from “green” or “amber”
to “red”. Alternatively, as in this example, a single date can be entered as the target for a group of tasks.

Setting a target date for this project’s tasks results in two of them having “red” status (which occurs when the 50% confidence end date falls after the target) and one with “amber” status (in this case just the 75% confidence end date is after the target). The team may wish to discuss with the stakeholders for this system whether they should delay the release date in order to complete these at risk features or whether it is more desirable to reduce the initial functionality to ensure the targeted release date is met. Often it is time rather than features that is at the highest premium, but planning in this way allows such discussions to occur early in the life-cycle and indeed continually as changes occur.

![Figure 7. Setting targets and the resulting task status icons](image)

Decision support of this kind is crucial for effective priority-based planning and in the design of priority-driven processes. Initial “what-if” modelling is important – for example considering the risks associated with particular tasks and their estimates – but it is equally important that the decision support remains in place as team members begin to carry out the work and possibly discover new work which is necessary before completion or propose new solutions to the problems faced. Agility is a crucial attribute required by all businesses today, but to achieve it teams must be able to adapt their schedules in the face of the constant change.

**Overhead (Date-based) tasks**

There is every possibility that this particular project would hit some serious overruns if the schedule was left at this stage. Looking at one of the team’s Personal Planners after
just the first couple of days of the project gives us a clue of a typical issue that teams must be aware of. Figure 8 shows part of the Personal Planner for one of the team members (Cath) for the first few days.

![](/image)

**Figure 8. Overhead tasks may take a lot of the available time**

Cath’s first main task is “Install Web Server” which is estimated to take between 6 and 10 hours. It is shown on the original schedule as due to complete on the first day. Her planner however shows that in fact she had only spent two hours on this task by Thursday of the first week, with the rest of the time taken up by meetings, administration and communications and a lot of other activity that, while it may be *predictable*, it is not what may be referred to as “plannable”. Such activity will continue throughout the project and if we do not take it into account the schedule will be wildly out. Neils Malotaux defines unplannable tasks as those activities that will take place whether you plan them or not! His estimate of the ratio of plannable to unplannable tasks on projects (based on actual project research) is around 55% [3]. Care must be taken with estimating though as sometimes inflated estimates are made to take these overheads into account. However if you want to base your estimates on “ideal time”, which is the most intuitive way of estimating tasks of up to a few days duration, then you must also include separate allowance for such unplannable tasks.

xProcess includes a special kind of task, the “overhead” (or “date-based”) task, for just these kinds of activities. It can also be used for meetings or appointments of known date and duration. Unlike “effort-based” tasks, overhead tasks have defined start and end dates which are independent of the amount of time booked to the task. As with prioritized tasks the dates may be modified by constraints, and rather than having estimates for the total effort required to complete the task, overhead tasks are specified as taking either a specified amount of time each day for each person assigned to the task, or a specified proportion of their time.
When these tasks are added into our schedule, the timescales of the project change considerably. (We have added two such tasks “Team Meetings” taking an average of 20 minutes a day per person, and “Team overheads” taking 20% of each person’s available time each day.) People’s availability on the project, and their ratio of overheads to plannable time, can of course be set on an individual basis where different team members are likely to have different proportions of unplanned activities.

Figure 9 shows the effect of these overhead tasks on the schedule. Many more tasks now have “red” status against our original target and as a consequence some of the essential features of the system are unlikely to finish on time. When situations like this arise it is important to get early-warning. The sooner bad news is heard and acted upon the more likely it is that commitments to customers and managers can be honoured or at least covered effectively. We will reset these targets in considering the final basic pattern yet to be used on this project, the timebox.

![Timebox example](image)

**Figure 9. Effect of the overhead tasks on the schedule**

**Timeboxes**

Timeboxes are examples of a general concept referred to in xProcess as “planning boundaries” or “folders”. They consist of a collection of tasks; they have target start and end dates and may optionally be prioritized. In other words a timebox is an agreed amount of work to be completed in a given time period. They are most useful for controlling the detailed planning of an iterative project.

Projects may have more than one periodic cycle: for example timeboxes of 2 week duration might be used by the team for their detailed planning of tasks, while release cycles of 60 or 90 days could be used for regular updates delivered to users. Agile methods such as
Scrum, eXtreme Programming and FDD use time-boxing techniques to ensure the team is focused on a well understood set of tasks for the immediate period (usually between one and four weeks). At the end of this period the team assesses its progress in completing the tasks identified and applies the lessons learned from that timebox in planning the next one.

Let’s revisit our example project a week after the Gantt chart shown in figure 9. This project has decided to use 2 week timeboxes so the first timebox is now complete and its second timebox, which runs from 15th January to 28th, is just beginning. The team starts each timebox with a meeting to agree the prioritized tasks to be completed in the period. In figure 10 we can see the target tasks that have been entered following the team meeting.

![Figure 10. The project’s second Timebox](image)

In figure 11 these tasks (with targets reset to correspond to forecast dates) are shown in Gantt chart form. Resetting the target dates to the 95% confidence end dates will result in the status icons on the tasks being green. Setting them to the 50% confidence end dates result in amber status.

---

1 Note the dates shown are in European rather that US format, a user-configurable option.
Figure 11. Gantt chart showing Timebox 2 tasks

We can also see that most of the tasks should be completed well within the two weeks, suggesting the team will probably be able to do some of the other outstanding tasks in addition to these tasks. This is why the final task (“Support feedback…””) has been added to the timebox even though the forecast end is after the end of the timebox (as shown by its red background). If things go better than expected the team will be able to deliver this feature too.

The work delivered after the first timebox – although only from the early stages of the project – provides the users and other stakeholders with an early view of the system. Feedback from them is likely to raise a number of issues, some of which may be accepted as new requirements and others as highlighting potential defects in the system or specification. The nature of priority-driven planning means that these tasks can be assessed for their business benefit against the existing tasks and, if recognized as higher priority, scheduled sooner than the other tasks. Timebox thus follows timebox, addressing the issues of highest priority to the business.

Another view that can help teams review their progress day by day against agreed targets is the burndown chart, which is supported in xProcess as a configurable report. An example is shown in Figure 12. This chart compares the target completion dates (usually set at the beginning of each timebox with the actual and forecast completion dates. When the lines diverge, in either direction the team gets early warning that things are not going quite to plan and can take the appropriate action or notify stakeholders of the situation. When new tasks are added or resources are changed during an iteration, the impact of the change can be immediately highlighted.
Enhancing the process

The example in this paper has explored the use of priorities of tasks as effective management control for agile projects. Having examined a project in this way we can draw lessons that could apply to projects of a similar type, and so build up the specific patterns that apply to all of them. This is what is required to define a process and we examine this in more detail in the following paper in this series [1]. At this point though it is useful to consider what elements make up processes in xProcess and which ones may be useful for the team in our example. The elements that can be defined in a process include:

- Project patterns
- Task patterns
- Artifacts and templates
- Tailored quality checking procedures (or Gateways).

If the team in this example had a project pattern set up that included the expected overhead task (such as the daily meeting and administrative overheads) a more realistic picture of the team’s schedule would have emerged immediately. Such a project pattern could also include the prioritized group of all project tasks that was used for prioritizing.

The main task pattern discussed has been the “feature” or new requirement task. Other than adjusting the default estimates for a new requirement, this team made no adjustment to the delivered task pattern. However there are a number of possibilities for enhancing the approach. There may be different types of “role” that are involved with new requirements (for example the specifier, developer and tester). If these roles are carried out by different team members, which in a larger team they certainly would be, separate tasks can be identified in the task patterns which can then be scheduled separately. There are
also a number of artifacts (or documents) which may be required by a more controlled process, such as an agreed statement of the requirement and an acceptance test specification. The templates for these documents can be defined in xProcess and the individual documents managed within xProcess’s distributed and versioned repository.

Task patterns may also be used for events such as defect identification, project risks and system releases. If appropriate, xProcess’s workflow server may be used to detect these events and instantiate the corresponding pattern. The patterns of tasks associated with review meetings for timeboxes can similarly be defined.

Finally the quality checking used by the team can be enhanced through the use of gateways – effectively a standard set of questions and audit checks that are required to be answered before tasks can be closed. The team in this case may wish to know that unit tests have been written (and pass), user documentation is complete and acceptance tests signed off. Answers are classified as “pass” or “fail”, and while a fail answer does not prevent a task from being closed, a rationale is required which allows inspection and reflection on the process to be carried out after the fact.

Conclusions

A process improvement environment such as xProcess is the ideal place to start in gaining control of the many projects in an organization, and the processes they follow. Projects can start planning activities – in particular planning by priority – before any specific process has been defined, and capture the patterns of working that will eventually form the basis of a fully defined priority-driven process. Such processes provide a convenient and agile mechanism for controlling fast moving projects. Combined with the live planning capability of xProcess, they ensure that changing priorities results in immediate changes to the planned work of team members and immediate visibility of the impact of the change, and indeed any other changes to resources, availability, targets and constraints.

Further Reading

This paper is the second in a series on Dynamic Process Management and xProcess being published by Ivis Technologies. Papers in this series address such topics as:

- Agility and transparency
- The structure of processes and how they are defined.
- The use of workflow and other generated actions within xProcess.
- Use of targets and forecasts for gap analysis
- The use of quality gateways.

Other papers in this series, as well as complementary materials and help with defining xProcess processes are available from [www.ivis.com](http://www.ivis.com). You can also provide feedback on issues raised in this paper at [xprocess.blogspot.com](http://xprocess.blogspot.com).
References


About Ivis Technologies

Ivis Technologies is a leading provider of software solutions to enable organizations to improve project planning and execution, while simultaneously improving their processes. Organizations that optimize processes and make the best use of resources, benefit with increased efficiency, quality and productivity. xProcess helps organizations become more efficient, productive and agile. It is an environment that brings together project management with project planning, forecasting and execution. xProcess helps skilled professionals collaborate to fulfill complex goals and to optimize plans, improve processes and manage resources.

Founded in 2002, Ivis Technologies’ products have been adopted by leading organizations in several industries. Ivis Technologies is headquartered in Phoenix, AZ with additional offices in Southampton, UK.

Ivis World Headquarters

5110 N. 44th St
Suite 200L
Phoenix, Arizona 85018
United States
Tel: +1 (602) 343-8200
sales@ivis.com
www.ivis.com

Ivis International Software (UK)

Westwood Centre
Nutwood Way
Totton
Southampton SO40 3SZ, UK
Tel: +44 (0)23 8087 5899
eurosales@ivis.com
www.ivis.com