Le projet **DELIRE**

Développement par **E**quipe

de **L**ivrables **I**nformatiques

et **R**éalisation **E**ncadrée

PStr7 – Planning et Budget des phases de réalisation et de convergence

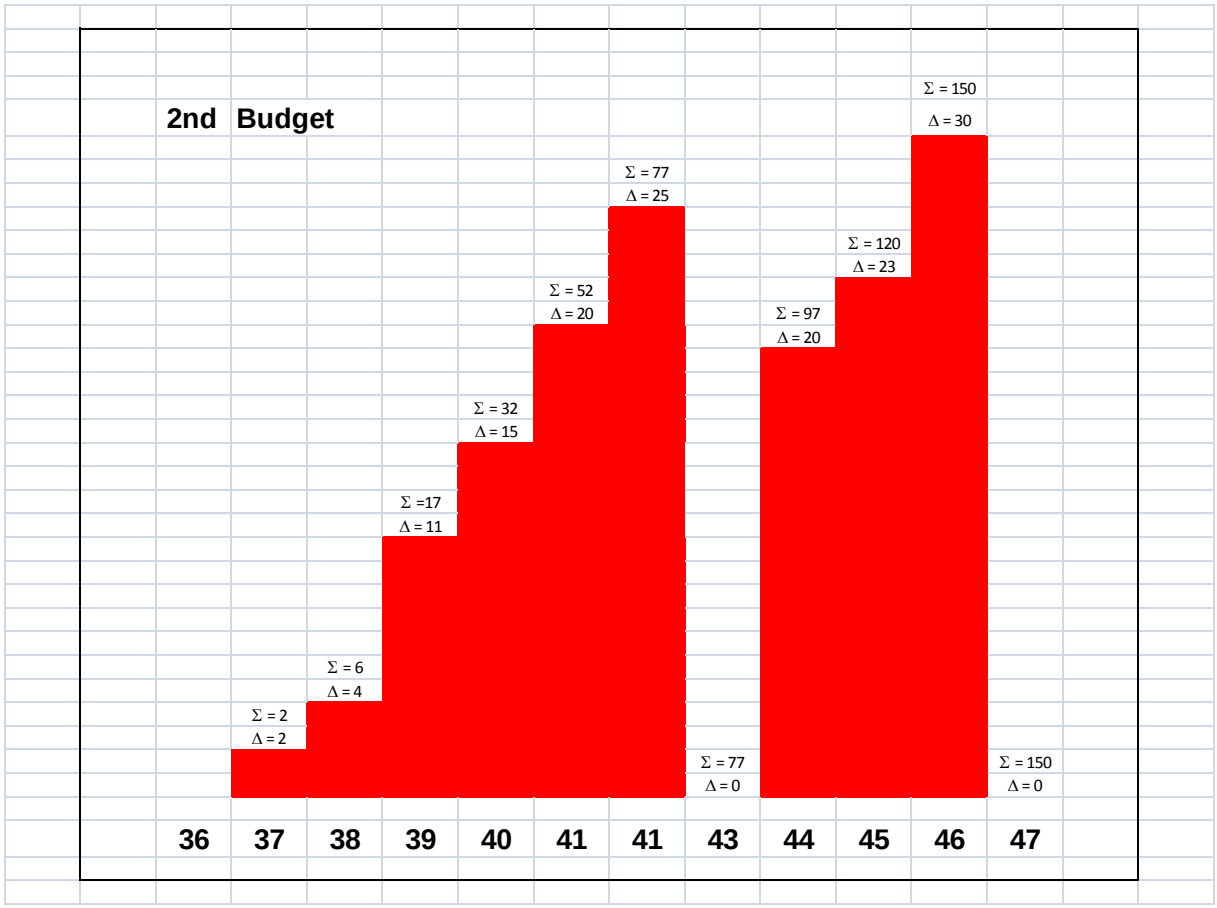
PLANNING ET BUDGET

Je rappelle que ce que j'appelle ici :

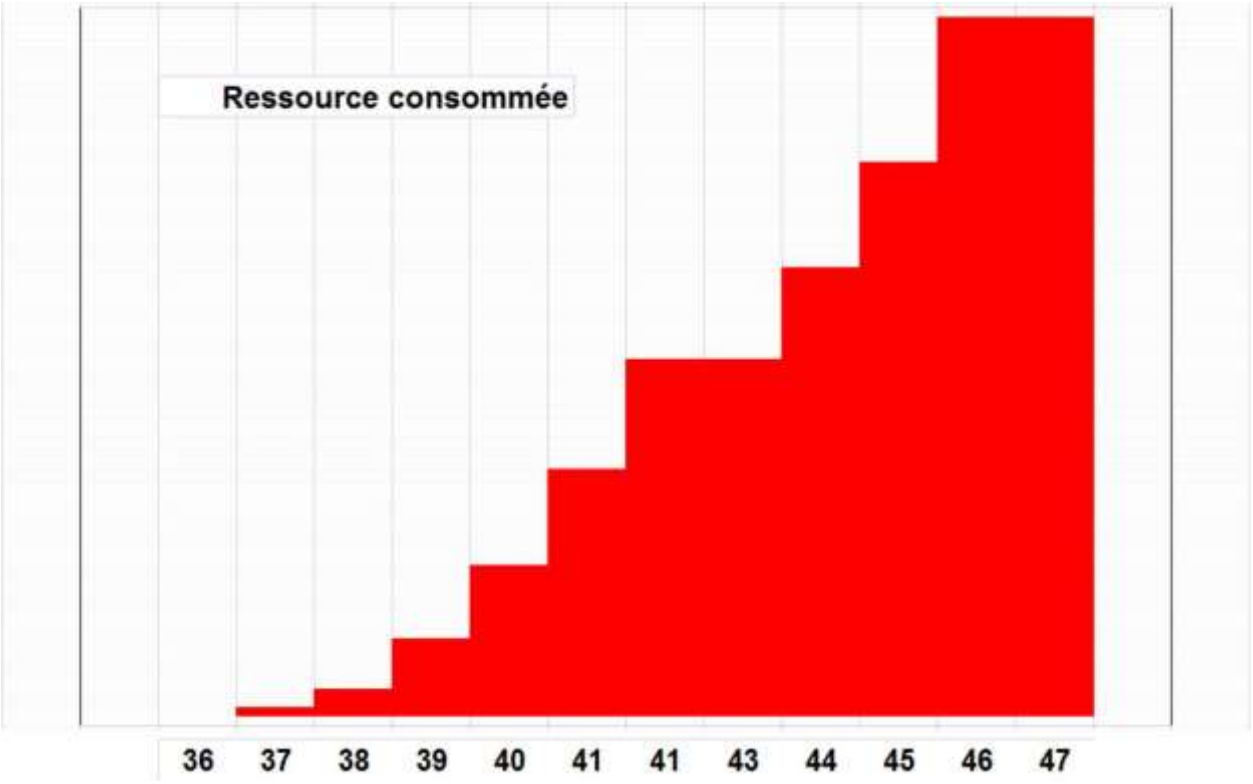
1. Planning est la définition d'un Gantt du projet. Pour chaque ressource (c'est-à-dire chaque membre de l'équipe) on identifie les activités f(t).
2. Budget est la consommation f(t) en nombre d'heures de votre projet.

Le planning se présente comme suit

Le budget se présente comme suit, en consommation = f(t)



ou comme suit en consommation sommée = f(t)



Remarque: ce qui vient d'être présenté n'est pas représentatif, puisqu'en effet j'ai représenté ici un planning et un budget de phase de spécification et de structuration.

Votre planning répond à 2 objectifs

1. Planifier de façon optimale
2. Communiquer sur le planning et les choix qu'il impose

Il vous permettra :

1. de déterminer les dates clé du projet;
2. d'identifier les marges existantes pour chaque tâche;
3. de visualiser le retard ou l'avancement des travaux.
4. De visualiser l'activité de chacun à un instant donné

Dans la gestion d'une entreprise, du fait qu'une ressource doit se partager entre plusieurs projets, ou plusieurs fabrication, le diagramme de Gantt est un outils trop frustre. Il est nécessaire de faire appel à des algorithmes plus complexes issus de la recherche opérationnelle et de la théorie de l'ordonnancement.

Dans le projet DELIRE, nous nous contenterons d'un diagramme de Gantt. Globalement, si le travail d'architecture a été bien fait, les dépendances entre tâche dans la phase de réalisation sont « relativement » faibles.
La question est donc de savoir comment ordonnancer les tâches durant cette phase. Il est souvent possible de trouver des solutions satisfaisantes en appliquant simplement des règles de priorité heuristiques. La méthode consiste à placer les tâches à effectuer dans le diagramme de Gantt dans l'ordre défini par la priorité et en tenant compte des ressources encore disponibles.

En production, les règles les plus courantes sont :

1. priorité à la réalisation des fabrications dont la date de livraison est la plus rapprochée ;
2. priorité à la première commande arrivée ;
3. priorité aux fabrications dont la durée totale est la plus courte ;
4. priorité aux fabrications qui utilisent le moins une ressource critique ;
5. priorité aux fabrications qui disposent du minimum de marge globale, c'est-à-dire qui sont sur le chemin critique.

La démarche

Vous venez de créer votre O.T.(Organigramme des Tâches), qui ne se limite pas aux seules tâches de coding, mais englobe la totalité des tâches du projet au second semestre :

1. Taches de coding
2. Taches de tests
3. Management et Suivi de projet
4. Rédaction de la seconde version des livrables
5. Préparation de la soutenance finale (présentation et démonstration)

Quel que soit le projet, il manque toujours une à deux semaines à la fin. Le mieux est de l'accepter, et de construire votre planning comme si votre soutenance se tenait deux semaines plus tôt que ce qui est réellement prévu.

La phase de coding est celle qui vous inquiète le plus. Dans la réalité d'un projet informatique, le cout des tests est au moins égal au cout du coding.

Je vous conseille de décomposer les 50% de travail du second semestre de la façon suivante :

1. 20% pour le coding
2. 20% pour les tests
3. 10¨% pour management et suivi de projet, rédaction des livrables et préparation de la soutenance.

Pour une équipe de 6 personnes (300 heures de travail), cela donnera

1. 120 heures pour le coding
2. 120 heures pour les tests
3. 60 heures pour management et suivi de projet, rédaction des livrables et préparation de la soutenance.

Comment dimensionner les tests ?
Dans le cout de tests, j'inclus l'exécution des tests, l'analyse des erreurs, la correction des bugs et la validation de la correction.
Comme vous l'avez (peut-être) lu dans le livre **PSS11 – Le P.T. (Plan de tests)** je vous propose de cibler sur 2 types de tests : des tests unitaires, et des tests Intégration/fonctionnels.
Pour mémoire je rappelle que les scénarios de tests ont été définis au premier semestre.

Faisons simple pour construire notre planning et notre budget :

1. Considérons que le coût des tests unitaires et le coût des tests Intégration et/ou fonctionnels sont globalement équivalents : 60 heures chacun
2. Considérons que si l'écriture d'un composant coute 4, l'écriture du test unitaire (qui se fait traditionnellement avant l'écriture du composant) coute 1, et l'exécution, l'analyse de l'erreur, sa correction et sa validation coutent également 1.

La difficulté est donc de dimensionner et de planifier les taches de programmation.

Positionner l'écriture du composant ou de l'atome de coding B après celle du composant ou de l'atome de coding A, sous le prétexte que B appelle A, n'est pas une nécessité : c'est tout l'intérêt de la phase d'architecture d'avoir figé les interfaces des composants ou des atomes de coding que de pouvoir appeler un composant qui n'a pas encore été écrit.
Le test d'intégration d'un composant ou d'un atome de coding présuppose que tous les composants qu'il appelle sont disponibles.
A contrario, en phase de tests unitaires, on réalise des bancs de tests qui simulent le comportement des composants autres que celui qu'on teste.

Depuis le 19ème siècle et les études de Norton Rayleigh, on sait que la courbe optimale pour un projet est une courbe grosso modo de Gauss (ou courbe en cloche). Cela est du au fait que la productivité maximale est établie lorsque chacun travaille de façon indépendante. Ceci arrive :

1. Lorsqu'on finalise les spécifications détaillées de chaque code source
2. Lorsqu'on commence à écrire les code source et les composants

Par contre le travail de spécifications générales et d'architecture nécessite un travail en commun. De même les phases d'intégration et de qualification du projet nécessitent un travail en commun.

Le projet DELIRE s'étale sur une vingtaine de semaines, et suppose une centaine d'heures de travail par personne : ceci représente 5 par semaine et par personne. La courbe de Gauss va ainsi prévoir que l'investissement de chacun est d'environ 2 heure par semaine en début et en fin de projet, Mais de 10 à 15 heures par semaine et par personne aux périodes les plus chargées : en fin de spécification et de structuration du projet, juste avant la soutenance de novembre, et en janvier au démarrage de la période de coding.

Un piège classique consiste à ne dimensionner que les macros tâches. Exemple : 120 heures pour le coding, 50 heures pour les tests d'intégration, 12 heures pour la construction de la présentation finale…
Puis on tartine les temps sur les différents composants : 30 heures pour la base de données, 25 heures pour l'interface user…
Le résultat est impeccable à la soutenance intermédiaire. C'est en général à la soutenance finale qu'on avoue que le planning n'a pas été respecté du tout : et généralement le coût a été beaucoup plus important qu'initialement prévu.

Pour être crédible, le cout de chaque tâche doit être fait à partir de paramètres de la tâche : nombre de lignes de codes, nombre de boucles et de branchement, nombre d'articles de menu, nombre de fenêtres à tester…

On compare ensuite le dimensionnement prévu pour chaque macro tâche par sommation des coûts élémentaires, et on compare à la valeur qui avait été dimensionnée.
Si le coût global du projet par sommation est très éloigné de la prévision initiale (et en général nettement supérieur), il est nécessaire de revenir sur le scope du projet : on réduit l'envergure de la demande, ou on augmente le délai et le coût. Dans le cadre du projet DELIRE, modifier le cout ou le délai est assez délicat. La bonne solution consiste à identifier plusieurs étapes de coding, et à se résoudre à ne pouvoir livrer qu'une partie du projet initial.

Très bien Antoine, mais encore faut-il connaitre les paramètres pertinents pour chaque tâche, et le coût que cela induit. Ce qui suppose d'avoir une bonne expérience. Oui certes.

Tout d'abord sachez que votre expérience n'est pas nulle : vous avez plusieurs années d'informatique à l'école ou en université. Sachez aussi que vous pouvez trouver des informations intéressantes sur le Net.
Mais l'essentiel est surtout d'intuiter des paramètres et des coûts, pour vous construire en fin de projet une expérience en validant, ou en invalidant, vos hypothèses initiales.

Il est classique également de ne dimensionner que les tâches nominales. Deux coûts sont en général oubliés :

1. Les tâches créées lors de l'analyse des risques
2. Le cout de management du projet. Il faut considérer que 5% à 10% du coût du projet pour gérer le planning, le suivi, la motivation de chacun ou la communication est une bonne estimation. En d'autres termes, chef de projet est un boulot, sinon à plein temps, en tout cas chronophage.

Dans un projet, il est classique d'avoir des dérives, c'est-à-dire de prendre du retard sur le planning nominatif, lequel est celui que vous produisez à la fin de la phase de spécification et de structuration, en vous appuyant sur l'O.T. Sachez-le, un retard pris ne se rattrape en général pas. Le mieux est donc d'anticiper les causes de dérapage. J'en citerai 5 majeures :

1. La première est de se tromper sur la complétude des tâches : c'est le rôle de la phase d'architecture et de construction de l'O.T. de minimiser ce risque.
2. La seconde est de se tromper le dimensionnement des tâches. En particulier, pour une tâche qu'on ne maitrise pas, on a souvent tendance à être optimiste sur son coût. Prévoyez ce fait dans votre gestion des risques et des buffers en conséquence, ce qui vous permettra de vous offrir quelques expériences sur un ou deux domaines d'innovation
3. La troisième est de découvrir que tout se passe mal. Sachez-le, c'est souvent classique dans un projet : tous les problèmes ont la fâcheuse tendance à s'accumuler, et comme on ne sait pas comment les résoudre, il y a une période de flottement pendant laquelle le dérapage s'accentue. C'est un autre des objectifs majeurs du PGR (Plan de Gestion des Risques) de vous forcer à préparer des plans de recouvrement en cas de problème grave. C'est également un des objectifs du PM (Plan Management) de vous préparer à savoir réagir en cas de tempête et de panique. .

4. Le quatrième est de découvrir qu'une ressource sur laquelle on comptait pour une tâche n'est pas disponible au jour J. Bien entendu cette tâche a le mauvais goût d'être située sur le chemin critique, et c'est tout le projet qui dérape. Un des rôles principaux du responsable de planification est de maintenir un planning à jour tout au long du projet. Il est bon également de dupliquer les savoir-faire.
5. La cinquième est spécifique au projet DELIRE : vous avez un temps mort pendant la période des fêtes. Et la reprise se révèle beaucoup plus difficile qu'escomptée. J'ai beau vous prévenir aujourd'hui, ce n'est pas pour cela que vous éviterez ce piège.

Un autre piège classique est de ne pas se souvenir d'un des principes du cycle en V : anticiper les problèmes de la phase suivante dans la phase en cours de réalisation. Ainsi :

1. On ne pousse pas suffisamment loin son architecture, en se disant qu'on traitera les problèmes durant le coding lorsqu'ils apparaitront
2. On néglige la gestion des risques : on ne va tout de même pas avoir tous ces problèmes
3. On remet à plus tard l'écriture des tests
4. On n'intègre pas en phase de coding les outils de debugging : mode trace, gestion des erreurs.
5. Pourquoi documenter son code, il est suffisamment explicite
6. Les standards de coding ont bien été documentés, mais néanmoins personne ne les applique.

Et finalement toutes ces petites compromissions court-termistes se révèlent source de dérapage en fin de projet.

Je vais enfin vous donner un conseil pour plus tard, lorsque vous serez dans l'industrie : ne pas se satisfaire d'une première solution Délai/Coût/Qualité pour votre projet. Optimisez-le ensuite, les gains ou les réductions de complexité peuvent parfois être impressionnantes. Je me doute néanmoins que le projet DELIRE, limité en délai et en coût, ne pourra être le cadre de ce type d'optimisation.

DERAPAGE D'UN PROJET

Le Viaduc de Millau est considéré comme un modèle de gestion de projet, malgré la difficulté technique du projet et le nombre d'acteurs impliqués

1) Spécification techniques tenues
2) Respect du budget
3) Respect du planning

Plusieurs points permettent de mieux comprendre comment ce programme a été géré, et les raisons qui ont fait ce succès :

1) Qualité du travail d'architecture, travail très long et très soigné de la structuration et de la planification du chantier
   a) 13 ans d'études techniques et financières.
   b) Pour 3 ans de réalisation
2) Très grand savoir-faire du maître d'œuvre
3) Le maître d'œuvre, Eiffage, était aussi est celui qui finançait le projet
4) Qualités humaines exceptionnelles de Marc Buonomo, le chef de chantier

Néanmoins, le Viaduc de Millau est une exception. Déraper dans un projet est normal. Plus de 85 % des projets dérapent, en particulier en informatique. Et ce, même lorsqu'ils sont gérés par des responsables aguerris.
Autant affirmer que vous avez toutes les chances de déraper. De toute façon, il faut l'avoir fait plusieurs fois dans sa vie professionnelle. Bienvenue au club.

Les causes de dérive, et en particulier de dérapage, d'un projet sont classiques, multiples…

1. Mauvaise définition des besoins
2. Spécifications / Exigences instables
3. Planning et/ou budget irréaliste, sous estimation des couts
4. Mauvais choix des fournisseurs ou des acteurs
5. Expérience insuffisante des équipes du domaine
6. Mauvaise gestion du projet
7. Mauvaise communication avec le client, la direction, les équipes métier, l'équipe de projet
8. Faible implication de l'équipe
9. Planification insuffisante des changements technologiques

… et non exclusives

Dans le cas du projet DELIRE, les deux principaux problèmes auxquels vous serez probablement confrontés sont :

1. Des spécifications incomplètes, dues à un travail insuffisant sur la définition de l'architecture et des contenus de chacun des composants
2. Un planning trop serré, du à une estimation optimiste des taches prévues et une inflation de taches oubliées.

Un autre risque pourrait également être une démotivation de l'équipe, potentiellement engendrée par une mauvaise gestion de projet. Bonne nouvelle, la structure et les conditions un peu particulières du projet DELIRE ne favorisent pas l'émergence de ce type de problème.

Les conséquences défavorables d'un dérapage sont:

1. Pour l'ouvrage
    a. Faible qualité des produits
    b. Développement d'un système erroné ou inutilisable
    c. Rejet du nouveau système par les utilisateurs
    d. Défauts non fonctionnels: sécurité, maintenabilité, efficacité, rentabilité…
    e. Coûts insupportables pour l'entreprise
2. Pour l'œuvre
    a. Exigences indéterminées ou irréalisables
    b. Indétermination des interfaces
    c. Difficulté d'intégration
    d. Incidence de l'échec du projet sur le fonctionnement de l'entreprise
3. Pour la planification
    a. Coûts imprévisibles pour le projet
    b. Retards dans la livraison des produits
4. Pour le suivi
    a. Accroissement des coûts du projet
    b. Défaut de participation des acteurs
    c. Déficience des tâches exécutées, en interne ou en externe
    d. Evolutions des exigences, en particulier dues au retard.

Pour DELIRE, ceci risque de se traduire par :

1. Produit incomplet
2. Tâches de tests insuffisamment voire non réalisées
3. Fiabilité du produit très médiocre
4. Documentation utilisateur et d'installation non fournie.
5. Livrables incomplets

Plusieurs raisons à cet état de fait :

1. C'est un projet ambitieux, plus important que ceux que vous avez été amenés à réaliser jusqu'à présent. Or la complexité et par voie de conséquence le coût d'un projet ont tendance à augmenter comme le carré de sa taille. Par contre, l'esprit humain a toujours tendance à extrapoler de façon linéaire
2. On a tendance, en planification, à ne prendre en compte que les tâches qui se voient, et à oublier toutes les petites tâches qui semblent négligeables, mais qui finissent par représenter une charge importante
3. Enfin, on dimensionne toujours une tâche en faisant l'hypothèse que tout se passera bien : il s'avère que ce n'est pas souvent le cas.

Déraper n'est pas en soit même dramatique, pourvu qu'on ait en tête quelques principes

1) Prévisions de livraisons étalées dans le temps
2) Le temps perdu ne se rattrapant jamais, il faut savoir faire des compromis et se prévoir des buffers, en temps et en ressources.
3) Il est important d'identifier au plus tôt les dérapages

Prévisions de livraisons étalées dans le temps

Structurer un projet en étapes, qui définissent le cycle de vie du projet, est une démarche classique, que tout le monde adopte.

1. En général on pense à faire spécification, puis structuration, puis réalisation
2. Par contre, on ne pense pas toujours à envisager plusieurs étapes de livraison dans la réalisation du projet.

Découper la livraison en plusieurs versions permet :

1. De pouvoir abandonner certaines étapes en ayant un produit acceptable en fin de projet
2. De pouvoir anticiper des retours de client, et pouvoir encore redresser la barre
3. De pouvoir éliminer des problèmes sur un code de taille raisonnable, et étaler la charge de tests et de correction
4. D'identifier rapidement des problèmes éventuels de performance
5. D'être sûr que toute l'équipe avance à la même vitesse et de ne pas lancer certains membres de l'équipe sur des tâches qui ne serviront finalement à rien.
6. C'est souvent en fin de projet qu'on s'aperçoit qu'on y arrivera pas, au moment où l'organisation du projet et l'architecture du produit sont telles que le résultat est en tout ou rien. Et au total : rien.

Le temps perdu ne se rattrape jamais

L'homme est irrémédiablement optimiste. Même si toutes les premières tâches ont montré qu'elles coutaient en moyenne 2 fois plus cher que prévu, un chef de projet a du mal à reconnaître que son projet est en péril.
Un chef de projet est comme un enfant devant une galerie de jouets : il veut tout et ne rien devoir choisir. Il imagine qu'une solution miracle va apparaître, et gommer tous les problèmes antérieurs. Ou bien que les tâches suivantes vont se passer mieux que prévu. Il va donc affirmer mordicus que les problèmes ne sont que passagers, et que le projet reste sur des rails et atteindra ses objectifs de qualité, de délai et de cout : ce que j'appelle de la langue de bois.
Ne rêvez pas : c'est toujours en fin de projet que les ennuis s'amoncellent.

Par défaut un dérapage ne se rattrape pas. Lorsque vous constatez un dérapage, vous devez remettre à jour les prévisions issues de la phase de structuration : vos spécifications fonctionnelles, votre planning, votre budget.
Ayant compris que le temps perdu ne se rattrape pas et que la langue de bois n'y fait rien, il va vous falloir choisir sur quel critère vous souhaitez lâcher du lest, en gros sur quel axe de qualité total vous allez faire des compromis :

1) Qualité : vous réduisez les fonctions livrées au client. Et ceci est d'autant plus facile que vous avez adopté la stratégie de prévoir plusieurs étapes de livraison pour le projet : Basic, Standard et Advanced..
2) Délai : vous décalez la date de livraison (impossible avec DELIRE) ou vous commencez à entamer vos buffers (ce qui revient à fragiliser votre planning)
3) Coût : vous augmentez les ressources affectées au projet (en l'occurrence, pour le projet DELIRE, chacun augmentera son nombre d'heures hebdomadaires sur le projet). Je le dis franchement, ce n'est pas nécessairement compatible avec vos autres contraintes d'enseignement.

Ces différentes approches ne sont pas antinomiques.

Et en phase de réalisation
Faites vos choix en termes d'objectifs qualité le plus tôt possible. Modifier les fonctions livrées conduit souvent à des modifications d'architecture, opérations dont on sait qu'elles sont sources de déstabilisation

Il est important d'identifier au plus tôt les dérapages
Plus tôt un dérapage est identifié plus les mesures, pour gérer ce problème, pourront être efficaces. Et tout particulièrement le fait de gérer l'impact sur les taches avales, et pouvoir réaffecter les ressources impactées, se retrouvant bien involontairement en chômage technique. .
Mais identifier un dérapage suppose deux principes :

1. Il faut mettre en place des indicateurs.
    a. Le coût de suivi de projet semble toujours inutile au départ, et on a souvent tendance à rogner sur la mise en place des indicateurs. Ce n'est que lorsque le projet est parti en sucette qu'on regrette ces économies de bouts de chandelle.
    b. Ceci étant posé, il est souvent facile de trouver des indicateurs simples, rapides à mettre en place et à mesurer, qui serviront d'avertisseur : lorsque le dérapage sera identifié, il sera temps si besoin de faire des analyses plus fines des problèmes.
2. Une bonne communication est sans doute la meilleure arme.
    a. Le premier qui sent qu'il dérape est en général celui qui exécute la tâche. Encore faut-il que la personne se sente encouragé à faire part de cette information.
    b. Ce qui suppose que le chef de projet accepte un dérapage sans en faire un drame : un dérapage est quelque chose de normal dans un projet. Un chef de projet n'est pas là pour traumatiser ses équipes, mais au contraire pour les sécuriser.

Il y avait une telle panique à oser avouer à Noël Forgeard qu'il y avait un problème dans la gestion du câblage de l'A380 que lorsque ce problème a enfin été connu, il était trop tard et une erreur de design s'est transformée en catastrophe industrielle.

Mes conseils :

1. Attaquez la phase de réalisation dès la rentrée de janvier. Je sais que c'est difficile, mais c'est un gage de réussite pour les phases de réalisation puis de convergence
2. Gardez vos buffers pour la fin : vous en aurez toujours besoin.
    - Au minimum 2 semaines en fin de projet
    - Prévoyez une consommation réduite de ressources en fin de projet
    - En pleine phase de réalisation, au moment où la consommation des ressources est la plus importante dans la courbe de Gauss, vous pouvez augmenter les ressources consommées. Pourquoi cette proposition à priori étonnante : simplement parce que les tâches sont relativement découplées.
3. Maintenez votre planning à jour.
    1. Quand vous identifiez une dérive, commencez par la comprendre (coup de flemme, difficulté technique, composants nécessaires non disponibles, oubli dans l'architecture…) puis mettez en place un plan de résolution. Reconstruisez ensuite votre planning, en n'oubliant pas que

4. En fin de projet, surtout pas de panique. Le rôle du chef de projet pour accompagner les dernières tâches est fondamental : il doit rassurer ses collaborateurs, montrer qu'il y a quelqu'un à la barre, et savoir prendre rapidement des décisions. Et vos buffers sont encore là pour parer à des problèmes éventuels.

Mais Antoine, si en phase de réalisation on constate que les tâches se réalisent plus vite que prévu ? Heureux ingénieurs, surtout ne changez rien à vos objectifs fonctionnels, mais augmentez vos buffers, et consacrez plus de temps à la qualité du produit.

Bref, on fait comment ?

1. Commencez par positionner sur votre calendrier le planning qui vous a été proposé en début du projet DELIRE : phase de réalisation, phase de convergence…
2. Identifiez le ou les chemins critiques
3. Faites pour chaque membre du projet une répartition des charges par semaines
   Par exemple : 18 – 18 – 18 – 15 – 12 - 9 – 6 – 3 – 1 – 0 – 0
4. Ensuite, tentez de répartir vos tâches en remplissant semaine après semaine le planning, en respectant les dépendances entre tâches, en respectant dans la mesure du possible l'affectation prévue de responsable pour chaque tâche, et en appliquant les règles heuristiques que vous aurez définies (et documentées) .