

ÉLIMINER  
LES GOULETS  
D'ÉTRANGLEMENT  
DU TEST

Maximiser la qualité des logiciels à travers un processus de test  
basé sur les exigences

## Sommaire

Présentation générale . . . . .	3
Crise de la qualité . . . . .	3
Erreurs de spécification des exigences . . . . .	4
Couverture de test inappropriée . . . . .	5
Présentation de l'approche RBT . . . . .	5
Tests précoces et récurrents . . . . .	5
Des tests rationnels plutôt qu'intuitifs . . . . .	6
Optique quantitative et d'amélioration . . . . .	6
Le processus RBT . . . . .	6
Qualité des exigences . . . . .	7
Conception de cas de test logiques . . . . .	7
Qualité des cas de test . . . . .	8
Qualité de la conception et du code . . . . .	8
Perfectionnement et exécution des tests . . . . .	8
Traçabilité . . . . .	8
Synthèse du processus RBT . . . . .	9
Implémentation de la méthodologie RBT avec les outils Borland . . . . .	9
Architecture de la solution RBT de Borland . . . . .	9
Revue des exigences en fonction des objectifs métier . . . . .	10
Détection des ambiguïtés dans l'expression des exigences . . . . .	10
Formalisation et structuration des exigences . . . . .	10
Revue multi-intervenant des exigences et cas de test . . . . .	11
Mapping des exigences sur les cas d'utilisation . . . . .	11
Génération et optimisation des cas de test . . . . .	12
Traçabilité entre exigences et cas de test . . . . .	12
Perfectionnement et exécution des cas de test sur le code . . . . .	12
Synthèse . . . . .	13
Services de consulting Borland – Processus RBT . . . . .	13
Conclusion . . . . .	14

## Présentation générale

Les directions informatiques et métier s'accordent à reconnaître que les entreprises capables de livrer rapidement et pour un coût réduit des logiciels d'une qualité supérieure et prévisible disposent d'un avantage concurrentiel majeur. Pourtant en pratique, les challenges de qualité associés aux livraisons logicielles demeurent largement irrésolus... En effet, en dépit d'investissements conséquents dans les outils d'automatisation des tests seule une infime proportion du portefeuille applicatif bénéficie d'une couverture de test suffisante et une grande partie des applications ne répond pas exactement aux besoins exprimés. Ces problèmes affectent l'expérience des utilisateurs et entraînent leur désaffection, de lourds travaux de reprise et, in fine, une perte de compétitivité pour l'entreprise.

Cette situation s'explique principalement par une tendance trop répandue des organisations de développement à considérer la qualité comme une préoccupation a posteriori - les opérations de vérification de qualité ne débutant qu'à l'issue de la phase d'écriture du code. À ce stade, les équipes de test ne disposent que d'un temps très limité pour valider l'application et cette étape est souvent vécue comme un véritable goulet d'étranglement ralentissant la mise en production. Dans ce contexte, il est difficile de certifier que les exigences sont correctement définies, d'élaborer des plans de test cohérents, de couvrir l'ensemble des besoins exprimés et de disposer d'une visibilité suffisante sur les différents aspects Qualité de l'application testée. Il en résulte logiquement des coûts supplémentaires et de multiples frustrations pour tous les intervenants.

Ce livre blanc présente les principes fondamentaux de l'approche du test basé sur les exigences – dite RBT pour Requirements Based Testing – dont la vocation est d'instaurer un processus de test radicalement différent présentant les caractéristiques suivantes :

- Couplage fort des tests avec les exigences applicatives exprimées
- Intégration des tests aux différentes activités du cycle de vie logiciel
- Couverture supérieure, quantifiable, systématique et économique des tests et des exigences

Sur la base de ces principes, l'approche RBT élimine les goulets d'étranglement précédemment évoqués en instaurant un processus qualité continu et non a posteriori.

Ce livre blanc explique également comment les solutions Borland permettent aux entreprises de surmonter les challenges propres à l'approche RBT.

## Crise de la qualité

Différentes études – notamment le « Chaos Report » du Standish Group – démontrent que depuis plusieurs décennies, les projets informatiques ont une forte propension à ne respecter ni les budgets ni les délais qui leur sont impartis. La médiocre qualité des applications est le facteur fondamental sous-tendant ces multiples échecs et exigeant de lourds travaux de reprise (reformulation des exigences, reconception, réécriture du code, etc.) qui rallongent d'autant les cycles de livraison et occasionnent des coûts supplémentaires substantiels. Pour réduire ce taux d'échec élevé, il est donc indispensable de mieux comprendre les raisons sous-jacentes à ces défauts de qualité.

De multiples études – confirmées par des constatations empiriques – démontrent que les deux raisons essentielles du manque de qualité des applications sont d'une part les erreurs de spécification des exigences et d'autre part une couverture insuffisante des systèmes de test. Les chapitres qui suivent sont plus spécifiquement consacrés à ces enjeux clés.

### Erreurs de spécification des exigences

Il peut paraître surprenant que malgré des tests complets et satisfaisants, un système logiciel ne réponde pas aux besoins des utilisateurs... La raison est pourtant simple : l'équipe de développement a travaillé sur des spécifications incorrectes des exigences.

En effet, pour les applications complexes, les exigences sont souvent négociées dans le cadre de deux dialogues simultanés, évoluant continuellement pendant tout le cycle de vie du projet et focalisés autour de deux questions clés : « Que devons-nous construire ? » et « Que pouvons-nous construire ? » La pertinence de ces discussions conditionne souvent la qualité finale de l'application développée.

Cependant, leur mise en pratique n'est pas toujours simple. Ainsi, une étude de James Martin<sup>1</sup> démontre que 56 % des défauts identifiés dans les projets logiciels sont introduits lors de la définition des exigences. Cette même étude précise que près de la moitié de ces défauts résulte d'erreurs d'expression, d'ambiguïtés ou d'un manque de clarté des besoins ; l'autre moitié étant imputable au manque d'exhaustivité des spécifications (c'est-à-dire à des omissions pures et simples).

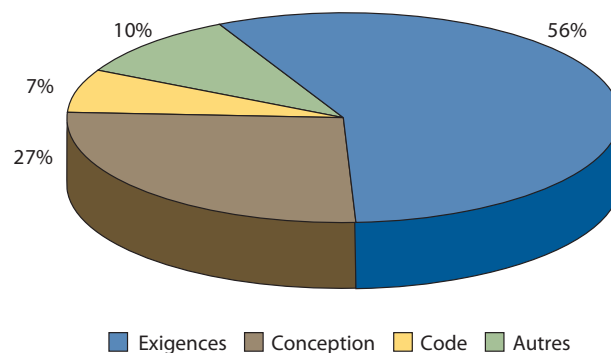


Figure 1 : Répartition des défauts affectant les projets logiciels

D'autres études révèlent des problèmes de même nature :

- 82 % des travaux de correction et de reprise sont liés aux erreurs de définition des exigences<sup>2</sup>
- Les problèmes de spécification des besoins représentent 44 % des causes d'annulation de projets<sup>3</sup>
- Seulement 54 % des exigences initiales sont effectivement réalisées<sup>4</sup>
- Seulement 45 % des exigences réalisées sont finalement utilisées<sup>5</sup>

### En synthèse, la qualité des exigences est entravée par deux problèmes fondamentaux :

- Le manque d'exhaustivité des exigences et spécifications en raison d'une implication insuffisante des utilisateurs et d'autres intervenants clés
- Des spécifications inadéquates des besoins par les experts du domaine, qui ne sont pas compétents pour définir des exigences cohérentes, précises et vérifiables et qui ne disposent pas d'outils efficaces pour valider et/ou examiner les exigences avec les utilisateurs.

<sup>1</sup> James Martin, « An Information Systems Manifesto »

<sup>2</sup> Martin & Leffinwell

<sup>3</sup> Rapport CHAOS - The Standish Group

<sup>4</sup> Rapport CHAOS - The Standish Group

<sup>5</sup> Jacobs

### Couverture de test inappropriée

La phase de test est une activité longue et fastidieuse exigeant une grande précision de planification et d'exécution. Le taux de couverture est en effet un facteur essentiel conditionnant la pertinence des tests et la capacité à déceler les problèmes importants comme ceux se posant dans le cadre de scénarii marginaux. Pourtant, même lorsque les exigences sont correctement formulées (complètes, exactes et vérifiables), il reste difficile de fournir une couverture optimale, et ce, pour les raisons suivantes :

- Les tests sont souvent conduits à la fin du processus de développement – au moment où les délais sont les plus serrés. Cette pression réduit d'autant la qualité de planification et d'exécution conduisant à limiter le pourcentage des fonctionnalités du système effectivement testées.
- En raison de la complexité des applications modernes (notamment des applications distribuées), il est très difficile de couvrir tous les scénarii – ne serait-ce qu'en raison de la multiplicité des « cheminements applicatifs ». Les approches consistant à envisager systématiquement (ou automatiquement) toutes les combinaisons possibles donnent lieu à des séries de tests ingérables ou trop longues à exécuter (qui retardent encore le processus global de certification).
- Les exigences applicatives changent fréquemment, mais ces changements ne sont pas correctement administrés. En raison de ce manque de traçabilité entre les exigences et les cas de test, il est difficile d'assurer une couverture continue des modifications.

Du fait de ces différents problèmes, les méthodes existantes de création et de sélection des cas de test sont la plupart du temps informelles et reposent essentiellement sur l'expérience et les compétences des équipes de test. Rarement encadrés, rigoureux et systématiques, les tests sont conçus sur la base d'intuitions et de ressentis ; autant de facteurs qui rendent la qualité logicielle relativement imprévisible.

## Présentation de l'approche RBT

L'implémentation de tests basés sur les exigences nécessite des processus parfaitement définis pour traiter les causes sources des problèmes de qualité évoqués précédemment. La méthodologie RBT repose sur les principes suivants :

- Réalisation de tests précoces et récurrents
- Méthodologie de définition rationnelle – et non pas intuitive
- Réalisation des tests dans une optique quantitative et d'amélioration

### Tests précoces et récurrents

La méthode RBT favorise un processus de test **intégré à toutes les phases du cycle de développement logiciel**. Dès que les exigences et les principes de conception et de codage sont définis, ils sont évalués et « testés » en fonction des bonnes pratiques, des objectifs métier, des exigences et des cas d'utilisation et des cas de test. Les tests sont menés parallèlement au processus de développement en impliquant toutes les fonctions de l'entreprise afin que tous les intervenants intègrent les objectifs de qualité.

Le processus de test n'est donc plus un goulet d'étranglement – intervenant après l'écriture du code et sous la pression des délais de livraison. Surtout, de nombreux défauts étant décelés en amont, leur résolution en est d'autant plus économique ; et les « surprises » à la livraison du code bien moins nombreuses...

### Des tests rationnels plutôt qu'intuitifs

Les tests basés sur les exigences privilégient une conception méthodique et systématique des cas de test pour maximiser la prévisibilité. S'appuyant sur un processus rigoureux, la méthodologie RBT ne repose pas exclusivement sur les compétences individuelles ou l'expérience des équipes de test mais sur un processus reproductible et méthodique de planification pour favoriser la prévisibilité de la couverture. Elle applique en outre différentes techniques d'optimisation visant à ne produire que le nombre minimal de cas nécessaires pour assurer une couverture suffisante et rendre le cycle de test plus rapide et administrable.

### Optique quantitative et d'amélioration

Le processus qualité de la méthodologie RBT est avant tout quantitatif et mesurable pour favoriser son administration et son amélioration constante à travers des critères quantifiables d'appréciation de l'état des modules livrables.

Les chefs de projets et experts des processus disposent ainsi d'une vision globale des initiatives qualité pour l'ensemble du portefeuille applicatif.

## Le processus RBT

Cette méthode a été conçue et affinée sur la base d'expériences pratiques et de recherches théoriques. À différents degrés, elle est largement implémentée dans les entreprises pour garantir que la définition des exigences et la conception des cas de test fournissent une bonne couverture de test et que les opérations de contrôle de qualité sont correctement intégrées à toutes les phases du processus de développement.

Le graphique ci-dessous détaille les activités et domaines stratégiques du processus RBT.

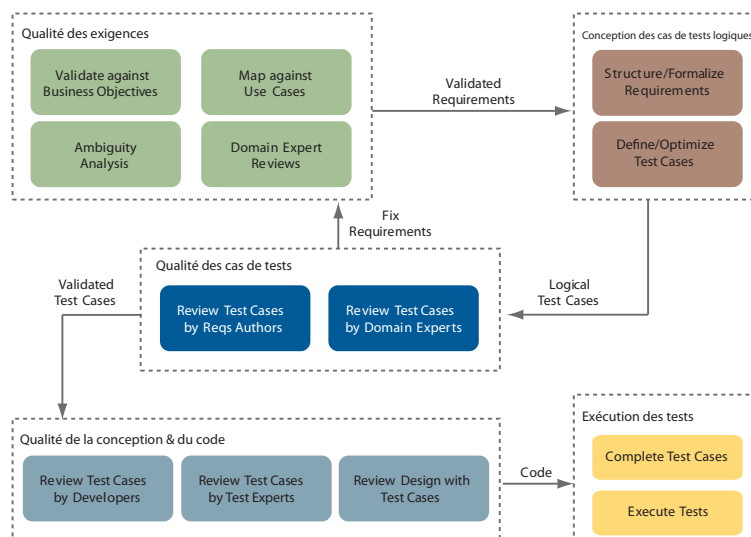


Figure 1 : Répartition des défauts affectant les projets logiciels

Contrairement aux activités de test « classiques » effectuées après la phase d'écriture du code, la méthodologie RBT préconise un processus de contrôle qualité tout au long du cycle de développement pour améliorer la pertinence des exigences et des cas de test et encourager leur utilisation dans les autres phases du cycle de livraison (conception, développement, etc.). Cette organisation garantit que les tests exécutés correspondent bien aux exigences métier exprimées à l'origine.

Les sections qui suivent décrivent plus en détail les différentes catégories d'activités du processus de test basé sur les exigences.

#### Qualité des exigences

Le succès des projets logiciels dépendant directement de la bonne compréhension et de la formulation précise des exigences, l'approche RBT accorde une importance particulière à leur qualité d'expression.

**Adéquation aux besoins métiers** – L'approche RBT inclut une étape de validation des exigences au regard des objectifs métier qui délimite clairement le périmètre du projet en garantissant que chaque exigence répond au minimum à un objectif métier. En cas d'inadéquation entre exigences et objectifs, une redéfinition est nécessaire.

**Implication des utilisateurs finaux et des experts de domaine** – La méthodologie RBT préconise une revue des exigences par ces intervenants et l'utilisation de leurs préconisations pour les ajuster si nécessaire avant d'aller plus loin.

**Validation de l'exhaustivité des exigences** – La méthodologie RBT recommande de les associer à une vue orientée tâches ou interactions du système (capturées à travers des cas d'utilisation) pour vérifier que les exigences répondent à tous les cas d'utilisation et valider leur exhaustivité.

**Cohérence et clarté des exigences** – La méthodologie RBT applique des techniques d'analyse linguistique afin de détecter les formulations problématiques et de les corriger. Une formulation problématique est une expression ambiguë, confuse ou incohérente pouvant être interprétée différemment en fonction des intervenants et générer des erreurs ; les termes ambigus affectent la testabilité des exigences.

#### Conception de cas de test logiques

L'expression textuelle des exigences a une importance primordiale, dans la mesure où c'est celle qu'utilisent « naturellement » les analystes métier et les experts du domaine pour les exprimer. Elle complexifie cependant la définition et la validation d'une couverture de test étendue. Lorsque les exigences sont exprimées en langage naturel informel, les équipes de test ne peuvent s'en remettre qu'à leur intuition pour valider l'exhaustivité de leurs cas de tests par rapport aux fonctionnalités identifiées dans les exigences. En d'autres termes, il n'existe pas de méthode formelle pour valider l'exhaustivité de la couverture et des scénarii marginaux – voire majeurs – peuvent « échapper » aux tests et ne se manifester qu'en phase de production.

Pour garantir une couverture systématique, l'approche RBT introduit une étape destinée à créer des représentations plus formelles et structurées des exigences. Il est alors possible de définir des cas de test logiques (ou « squelettes ») pour garantir une couverture optimale des exigences ; ces derniers seront ensuite ajustés pour créer les véritables jeux de tests exécutés sur le système.

De multiples méthodes peuvent être utilisées **pour structurer et formaliser des besoins exprimés en langage naturel**. Leur objectif est de révéler les relations de cause à effet incorporées dans les exigences, c'est-à-dire d'exprimer une exigence comme un ensemble de conditions (causes) et d'actions consécutives (effets). La présentation graphique des relations de cause à effet est l'une de ces méthodes. Une autre approche consiste à exprimer les exigences sous forme de diagrammes de flux représentant les dépendances entre actions et la ramification conditionnelle des activités.

Cette expression structurée permet ensuite de **déduire simplement les cas de test équivalents**. Un ensemble de cas de test logiques parfaitement « équivalent » aux fonctionnalités identifiées dans les exigences peut alors être défini (manuellement ou automatiquement). Il peut cependant inclure de nombreux cas redondants (chevauchements, etc.).

**Pour optimiser le nombre de cas** – tout en assurant une couverture élevée – différentes techniques (tableaux

décisionnels, etc.) peuvent être appliquées si la structuration des exigences a été réalisée à travers des diagrammes de cause à effet. En cas d'utilisation de diagrammes de flux, l'optimisation du nombre de cas impliquera d'identifier tous les chemins d'accès uniques – pour lesquels des techniques existent.

#### Qualité des cas de test

Il existe une probabilité que des erreurs subsistent dans les cas de test après leur conception. **Pour détecter et corriger ces erreurs**, la méthodologie RBT prévoit des revues des cas par les rédacteurs des exigences, les utilisateurs finaux et les experts de domaine. Cette étape donne à ces intervenants cruciaux l'opportunité de réexaminer les exigences et les cas de test sous leur forme structurée et de déceler les défauts apparus lors de la traduction du langage naturel en expression formelle.

#### Qualité de la conception et du code

La méthode RBT dépasse les approches traditionnelles en intégrant le processus qualité au développement à travers des cas de test structurés, validés et performants, générés lors des phases précédentes.

**Pour garantir la robustesse de la conception et son adéquation aux exigences**, ses principes sont analysés en fonction des cas de test logiques (qui ne sont qu'une représentation différente des exigences). Si une inadéquation est constatée, cela signifie que les exigences restent problématiques (et doivent être précisées) ou que la conception doit être revue.

**Pour garantir une bonne compréhension des éléments à tester par les développeurs** et leur fournir une vue structurée des exigences, l'approche RBT recommande de soumettre les cas de test à la validation des développeurs chargés du codage.

**Pour garantir la conformité du code aux exigences**, l'approche RBT recommande de revoir les modules individuels de code sur la base des exigences structurées correspondantes. Il est en effet bien plus simple de comparer les expressions algorithmiques du code à la vue formelle des exigences qu'à une expression non structurée.

#### Perfectionnement et exécution des tests

Les cas de test logiques peuvent désormais être complétés en définissant les entrées de données et les flux de navigation – et éventuellement en mettant en œuvre des outils d'automatisation. **Pour comparer le comportement réel du système à son comportement théorique**, l'ensemble des tests définis peut désormais être appliqué au code.

#### Traçabilité

Bien que n'étant pas, à proprement parler, une « phase » du processus RBT, la traçabilité contribue largement à son succès. Il est en effet crucial de maintenir la traçabilité entre les exigences, les cas de test et les tests proprement dits. Ces informations sont nécessaires pour contrôler l'avancement et la couverture des tests et pour gérer correctement l'impact des changements apportés aux exigences. Dans le cas contraire, il est très difficile de déterminer quels tests ou cas de test modifier en cas de changement d'un besoin spécifique.

## Synthèse du processus RBT

Le tableau suivant synthétise les objectifs, les finalités associées et les moyens de les atteindre prônés par la méthodologie RBT.

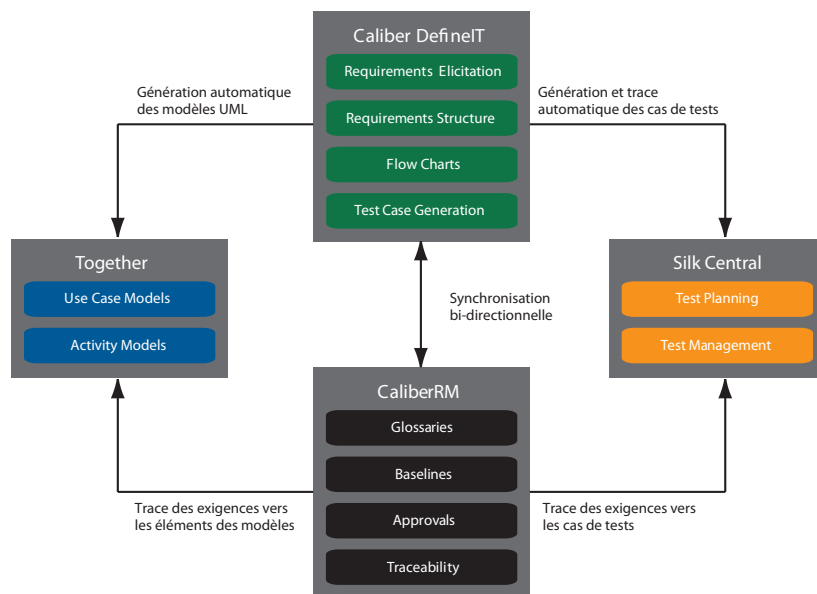
Objectifs	Finalités	Méthodes
Assurer la qualité des exigences	Garantir que les besoins métier sont effectivement satisfaits, favoriser l'implication des utilisateurs finaux et experts de domaine, valider l'exhaustivité des exigences et garantir leur cohérence et leur clarté	Revue des exigences par rapport aux objectifs métier par les experts de domaine et utilisateurs métier, mappage des exigences aux cas d'utilisation, analyse des ambiguïtés
Concevoir des cas de test logiques	Structurer et formaliser les exigences exprimées en langage naturel, déduire des cas de test équivalents, optimiser le nombre de cas de test	Expression des exigences à l'aide de graphiques de cause à effet ou de diagrammes de flux, génération automatique ou manuelle de cas de test, tableaux décisionnels ou détection des chemins d'accès uniques
Garantir la qualité des cas de test	Détecter et corriger les erreurs des cas de test	Revue des cas par les auteurs des exigences, experts de domaine et utilisateurs finaux
Garantir la qualité de conception et du code	Assurer une conception suffisamment robuste pour satisfaire les besoins, garantir que les développeurs comprennent les éléments à tester et valider que le code livré est conforme aux exigences	Revue des cas par les concepteurs et développeurs, utilisation de cas de test lors de la revue du code
Compléter et exécuter les tests	Comparer le comportement réel du système à son comportement attendu	Définition complète des cas de test, exécution sur le code livré

## Implémentation de la méthodologie RBT avec les outils Borland

Borland prend en charge le processus RBT à travers une large gamme de produits, de services professionnels et de prestations de formation. Ces technologies proposent un support étendu grâce à des produits simplifiant la collecte, la validation, la gestion et la traçabilité des exigences et le développement de cas de test pour les vérifier et les valider. Les programmes de formation Borland permettent par ailleurs aux équipes de développer leurs compétences en matière de planification des activités de contrôle de qualité, d'organiser des revues performantes des produits de travail et d'utiliser dans les meilleures conditions les suites technologiques Borland. L'offre de consulting Borland aide les organisations à identifier les lacunes majeures de leurs pratiques de gestion continue de la qualité, à implémenter itérativement des processus appropriés pour les combler, à développer les compétences requises et à intégrer des technologies à l'état de l'art à tous les stades du processus.

### Architecture de la solution RBT de Borland

La technologie de définition et de gestion des exigences de Borland applique la plupart des concepts fondamentaux de l'approche RBT : Caliber DefineIT permet d'explicitier et de structurer les exigences et de générer et optimiser les cas de test qui en découlent ; CaliberRM prend ensuite en charge l'administration des demandes de changement et leur traçabilité tout au long du cycle de vie et enfin l'intégration des outils de gestion des exigences aux solutions Borland Together et SilkCentral Test Manager permettent d'impliquer efficacement les concepteurs et les équipes de test au processus RBT.



Les chapitres suivants expliquent comment automatiser des parties fondamentales du processus RBT grâce aux produits et systèmes d'intégration Borland.

#### Revue des exigences en fonction des objectifs métier

Borland Tempo favorise la collaboration entre équipes informatiques et métier en assurant la collecte et la gestion des objectifs métier et le déploiement d'un processus performant de sélection et d'initialisation des projets. Les offres de formation et d'amélioration des processus de Borland fournissent une assistance pour collecter et exprimer les besoins fonctionnels, les objectifs métier et les exigences système. Après avoir articulé les exigences essentielles, les praticiens de la méthodologie RBT peuvent utiliser la plate-forme Tempo pour les analyser à la lumière des objectifs métier convenus.

#### Détection des ambiguïtés dans l'expression des exigences

Pour aider les rédacteurs des exigences à détecter et corriger les ambiguïtés, Borland propose des ateliers de développement des compétences de revue collaborative pour examiner précisément les artefacts de travail (exigences, conception, code, cas de test, etc.) et éliminer les défauts et les travaux de reprise du cycle de développement. Pour certaines équipes, des revues très formelles (inspections) sont nécessaires ; pour d'autres, de simples discussions ou examens techniques seront plus adaptés. Les ateliers Borland aident les équipes à déterminer quand planifier et exécuter ces différents types d'examen.

En outre, CaliberRM offre la possibilité d'inclure des glossaires pour aider les développeurs et réviseurs à minimiser tout risque d'ambiguïté et permettre aux analystes métier de définir de multiples dictionnaires incluant les termes considérés comme « inadaptés » et ceux devant être évités lors de la définition des exigences (parce que trop généraux ou ambigus). Il est également possible de définir des termes communs fréquemment utilisés dans les exigences pour favoriser une compréhension claire et homogène par l'ensemble des intervenants. Après la définition des glossaires, CaliberRM surligne automatiquement les segments de texte correspondant à des termes des glossaires et affiche des infobulles.

#### Formalisation et structuration des exigences

Borland Caliber DefineIT aide les analystes métier à structurer les exigences en les capturant lors des processus de définition ou d'explicitation. DefineIT structure les exigences sous forme de scénarii définissant des interactions de haut niveau avec l'application et comportant plusieurs étapes – chacune représentant une action à forte granularité du scénario.

DefineIT propose deux représentations structurées de chaque scénario : une représentation « textuelle » et une représentation graphique (diagramme de flux représentant la séquence d'étapes du scénario et sa logique de ramification).

Ces deux représentations sont synchronisées à tout moment (les actions effectuées dans une représentation sont immédiatement répercutées dans l'autre).

Les exigences étant structurées dans DefineIT, il est possible de valider leur cohérence logique. En effet, DefineIT propose un panneau de validation affichant des messages sur les problèmes logiques potentiels d'un scénario.

Par exemple, si un point décisionnel est ajouté à un scénario et que toutes les branches conduisent à la même étape, le point décisionnel est incohérent. DefineIT le notifie alors à l'utilisateur, garantissant ainsi une définition exacte et logique du scénario. Le panneau de validation est mis à jour en temps réel, à chaque manipulation du scénario par les utilisateurs.

#### Revue multi-intervenant des exigences et cas de test

Le processus RBT inclut des sessions de revue des exigences et des cas de test par les différents intervenants (experts de domaine, utilisateurs finaux, concepteurs, développeurs, etc.) qui fournissent à tous des informations claires et concises (techniques et non techniques) pour comprendre le comportement prévu du système. Leurs réactions et commentaires sont alors collectés et utilisés pour améliorer et affiner les exigences. Les ateliers de revue collaborative de Borland développent les compétences de tous les intervenants pour maximiser l'efficacité de ces sessions de revue.

L'outil DefineIT est principalement conçu pour éliminer les « écarts linguistiques » entre les départements informatiques et fonctionnels en structurant les échanges entre ces deux entités. En effet, les intervenants métier sont des experts du domaine mais ne maîtrisent pas forcément la terminologie technique ; à l'inverse, les équipes informatiques connaissent parfaitement le vocabulaire technique mais cernent moins bien les problématiques métier. DefineIT synchronise les points de vue en favorisant les transferts de compétences d'une équipe à l'autre.

Les fonctionnalités d'explicitation de Caliber DefineIT sont conçues pour définir précisément les exigences exprimées par différents groupes d'intervenants dans un processus de revue solidement structuré. Après la modélisation du scénario dans DefineIT, sa fonction de « story-board » permet de démontrer l'application à ses utilisateurs finaux et de valider son adéquation à leurs attentes.

En substance, la fonctionnalité de story-board de DefineIT consiste à accompagner les réviseurs sur de multiples « chemins » définis par les étapes et points décisionnels du scénario et à collecter leurs commentaires, qui seront utilisés en retour pour l'améliorer.

#### Mapping des exigences sur les cas d'utilisation

L'intégration des applications CaliberRM et DefineIT à la solution de modélisation Borland Together favorise la collaboration entre analystes métier et architectes pour s'assurer de la synchronisation des exigences avec les modèles de cas d'utilisation et pour garantir que la conception applicative repose sur des exigences validées et précisément définies.

Grâce à cette intégration entre DefineIT et Together, les utilisateurs peuvent exporter des projets DefineIT dans Together sous forme de diagrammes UML de cas d'utilisation et diagrammes d'activités pour transformer les représentations des scénarii DefineIT en modèles UML équivalents. Ces diagrammes incluent des liens aux documents de référence de DefineIT (images, etc.) directement accessibles depuis Together.

Il est également possible d'exporter des projets DefineIT sous forme de diagrammes BPMN (Business Process Modeling Notation) à travers les spécifications QVT (Query View Transfer) de l'OMG. Avec DefineIT, les utilisateurs peuvent exporter des projets sous forme de documents XMI, qui seront transformés par les architectes en diagrammes BPM grâce aux fonctionnalités QVT de Together. Enfin, Together permet de générer le langage d'exécution BPEL à partir du diagramme BPMN.

#### Génération et optimisation des cas de test

Caliber DefineIT est capable de générer automatiquement un jeu de cas de test pour un scénario spécifique d'un projet afin de maximiser la couverture des tests fonctionnels de l'application et minimiser le nombre de cas de test générés (chacun représentant un chemin unique du scénario). Lorsqu'un cas de test spécifique est sélectionné, l'application surligne le chemin associé dans le diagramme du scénario.

DefineIT génère une description textuelle de toutes les étapes du cas de test en spécifiant, pour chacune : l'intervenant concerné, la mesure requise et le résultat escompté. Les cas de test générés peuvent être exportés dans Borland SilkCentral ou en HTML, pour créer un site Web interactif permettant aux équipes d'assurance qualité de visualiser les cas de test et leurs artefacts associés.

#### Traçabilité entre exigences et cas de test

Les solutions CaliberRM et Caliber DefineIT permettent d'associer les cas de test aux exigences pour vérifier que les séries de test couvrent bien l'ensemble du périmètre applicatif. Cette fonctionnalité est renforcée par des intégrations avec SilkCentral Test Manager.

Grâce à l'intégration de DefineIT à SilkCentral, il est possible de transférer automatiquement dans ce dernier les exigences et cas de test générés par DefineIT. Les exigences exportées apparaissent alors dans l'arborescence du projet sélectionné, accompagnées des cas et étapes de test correspondants. Lorsque l'exportation est terminée, les cas de test sont automatiquement reliés à leurs exigences respectives dans SilkCentral.

CaliberRM offre les mêmes intégrations avec SilkCentral et de puissants outils de visualisation et d'analyse de traçabilité.

- Matrice de traçabilité – Puissante table interactive présentant sous forme de matrice l'ensemble des liens entre les exigences, les éléments du modèle, le code et les tests.
- Diagramme de traçabilité – Outil interactif offrant une représentation graphique des dépendances entre les besoins et les autres éléments du cycle de vie (ressources de test, etc.).

CaliberRM signale visuellement tous les liens « suspects » provenant ou émanant d'exigences modifiées. Cette fonctionnalité, particulièrement cruciale pour les analyses d'impact, permet aux équipes de test d'identifier et d'accéder rapidement à l'ensemble des artefacts de test affectés par une modification des exigences. Elles peuvent ensuite afficher et modifier tous les cas de test dans SilkCentral pour les ajuster en fonction de la modification apportée.

#### Perfectionnement et exécution des cas de test sur le code

Lorsque les « squelettes » des cas de test générés dans DefineIT sont exportés dans SilkCentral, il est possible de générer des définitions détaillées des tests et d'exécuter une série de tests fonctionnels et de performance en utilisant conjointement SilkCentral, SilkTest et SilkPerformer. Si les principes de test RBT ont été appliqués, l'équipe de test est assurée, à ce stade, que la couverture de test est optimale.

## Synthèse

Le tableau ci-dessous présente les correspondances entre les produits et services Borland et les composants RBT requis.

Composant processus RBT	Produit Borland correspondant
Collecte et documentation efficaces des exigences (à tous les niveaux)	Ateliers de collecte, définition, administration et ingénierie des exigences
Revue des exigences par rapport aux objectifs métier	Tempo, Ateliers de revue des exigences, CaliberRM
Revue des exigences par les experts du domaine et les utilisateurs fonctionnels	Atelier de revue collaborative des exigences, Caliber DefineIT / Explicitation
Mapping des exigences sur les cas d'utilisation	Intégrations de CaliberRM et DefineIT à Together
Analyse des ambiguïtés des exigences	Atelier de revue collaborative des exigences, glossaires CaliberRM
Expression formelle des exigences à travers des diagrammes de cause à effet ou de flux	Caliber DefineIT / diagrammes de flux
Génération automatique ou manuelle de cas de test	Caliber DefineIT / génération automatique de cas de test ; formations SilkTest et SilkPerformer
Optimisation du nombre de cas de test à l'aide de tables décisionnelles ou d'outils de détection des chemins uniques	Caliber DefineIT / diagrammes de flux ; formations SilkTest et SilkPerformer
Revue des cas de test par les rédacteurs des exigences, les experts du domaine et les utilisateurs finaux	Atelier de revue collaborative des exigences, Caliber DefineIT / Explicitation
Revue des cas de test par les concepteurs et développeurs	Atelier revue collaborative des exigences, Caliber DefineIT / Explicitation, formations SilkTest et SilkPerformer
Perfectionnement et exécution des cas de test sur le code livré	Gamme de produits et formations Borland Silk
Traçabilité entre exigences et cas de test	Intégrations de CaliberRM et DefineIT à SilkCentral

## Services de consulting Borland – Processus RBT

Les solutions Borland de gestion du cycle de vie qualité sont sous-tendues par une large gamme de prestations de formation et de développement de processus et par des technologies exclusives. La solution spécifique répondant aux caractéristiques propres à chaque client dépend de son niveau de maturité en matière de gestion du cycle de vie qualité et de ses objectifs. Borland utilise le cadre méthodologique « Accelerate » pour concevoir et livrer ces solutions personnalisées.

L'offre Borland de gestion de la qualité tout au long du cycle de développement inclut les composants suivants :

- Ateliers exécutifs (plusieurs heures) – Pour aider les décideurs à analyser les meilleures pratiques de l'industrie et à fixer les objectifs métier de leur organisation en matière d'amélioration de l'infrastructure de gestion du cycle de vie Qualité.
- Évaluation/analyse des écarts – Étude approfondie des processus, compétences et technologies existants par rapport aux meilleures pratiques de l'industrie pour identifier les domaines majeurs d'amélioration ; la durée dépend du nombre de domaines examinés et de la taille de l'organisation (en général, 3 à 10 jours)

- Atelier Planification action (3 jours) – Destiné à l'équipe en charge de l'implémentation des recommandations résultant de l'analyse d'écart.
- Consulting – Prestations de conseil et recommandations pour le développement et le déploiement des améliorations des compétences, processus et technologies ; assistance à la mise en œuvre des changements organisationnels.
- Atelier Revue collaborative des exigences (2 jours) – Présentation des différents types de revue des produits de travail et des méthodes de sélection (en fonction de la finalité) incluant des exemples pratiques d'inspections formelles.
- Formation sur les produits de la gamme Silk Testing et les pratiques associées – Offre complète de formation à l'utilisation des solutions Silk Central Test Manager, Silk Test et Silk Performer

## Conclusion

En intégrant les activités du processus qualité au cycle de développement global, la méthodologie de test basé sur les exigences garantit la prévisibilité du niveau de qualité logicielle. Grâce à l'approche RBT, la qualité retrouve l'importance qu'elle mérite et les activités de test ne sont plus perçues comme un goulet d'étranglement intervenant à l'extrême fin du projet. Les solutions et services de consulting Borland – utilisés isolément ou en accompagnement de solutions tierces de gestion des tests – fournissent une assistance précieuse dans les déploiements de processus de test orienté sur les exigences.