Web Services

Bertrand Blanc
ESSI – SAR 6
E-mail: bertrand.blanc@essi.fr

Table des matières

Présentation

Dans le contexte du module SAR6 *Internet et technologies Web* nous avons été amené à proposer une spécification d'un service web appliqué à un projet concret relatif à l'ESSI et à ses étudiants. Les trois principaux buts de ce projet sont: tout d'abord appréhender l'utlité et l'utilisation d'un modèle conceptuel de spécifications de *web services* en WSDL sur XML. Le second, moins axé technologies de l'Internet, permet de proposer des spécifications d'un service tel qu'un client pourrait le demander en fournissant un cahier des charges. Le troisième permettra de réaliser à long terme une implantation du service.

Ce document propose dans une première partie une description des *web services* au travers de cas d'utilisateurs. Dans la seconde partie, nous présenterons une application Internet sujette à une description de services web. Le sujet porte sur les phases de notification de *bugs* souvent laissées de côté faute de temps. La dernière partie est consacrée à la définition en WSDL de quelques éléments significatifs de l'application afin de proposer une architecture logicielle.

Ce sujet est d'un intérêt certain car ces phases de dépistage des erreurs sont primordiales dans le cycle de vie d'un logiciel. Le cycle de vie de référence choisi est le cycle en V. Le travail exposé ici porte sur les spécifications informelles, formelles, d'architecture et sur une partie de celles d'implantation. Le fond du sujet traite des phases de tests unitaires, d'intégration, de portage et de tests fonctionnels. Dans le cas où le sujet soit sélectionné, l'implantation devra en être faite. Ainsi, tout le cycle en V de développement du logiciel est ici passé en revue.

1 Les *web services*

1.1 Caractérisation des *web services*

Le document [2] décrit au travers de scénarii de cas d'utilisateur la manière dont nous souhaitons qu'un service web soit décrit dans la vie réelle. Les 30 cas d'utilisateurs exposé dans cet article présentént les spécifications d'un tel service, mettant ainsi en lumière une ou des architectures potentielles particulièrement adaptées aux services web.

Les applications mettent à disposition des interfaces de services publiant des messages *one-way* sur un canal de

messagerie logique c'est à dire indépendant du médium choisi. Lorsqu'un message est envoyé, l'application ne se préoccupe pas de recevoir un quelconque acquittement (*fire-and-forget*) mais désire cependant que de façon transparente, dans le cas où une erreur subvienne, en être informée. Ainsi, un service d'exception *multiple faults* est mis à disposition de l'application. Néanmoins, des requêtes avec accusé de reception doivent pouvoir être formulées.

La diversité d'utilisations offertes doivent prédire le potentiel d'expressivité des services web, avec en l'occurence prévoire un système de déclaration de services transactionnels pour les banques. De plus, dans une optique génie logiciel, la possibilité de déclarer une interface sans y attacher de code doit être laissée aux utilisateurs de sorte à dissocier les phases de spécifications et d'implantations.

L'échange des données est basé sur un échange asynchrone de documents *XML-like* ou pas (données numériques provenant d'une acquisition de caméra par exemple), sérialisation, ... Ainsi, la transmission d'un message doit être précédé d'une phase de construction d'un document joint qui résume l'information que l'opération attachée au service devra processer. Un service devra être capable de manipuler plusieurs styles de données indépendament du format (langages, impression, images, ...), de la taille (jusqu'à plusieurs centaines de mega-octets) ou du système hôte (PDA, téléphone cellulaire, ...). De plus, les services — qui peuvent être vus comme des composants — doivent pouvoir coopérer entre eux, en l'occurence pour l'authentification, l'encryption, la maintenance, ... Les moyens de d'échange de messages se réduisent à la requête et à la réponse (outre le mécanisme d'exceptions).

Un service web doit laisser la possibilité de traiter des types de données complexes tels que l'appel à un comportement via une opération attachée à un autre service web typant cette donnée. A ce niveau apparait le pouvoir d'expressivité dont le language de description des services web (WSDL) nécessite indépendament du système de transmission des messages: SOAP via plusieurs architectures de communication. Il apparait de plus que l'utilisateur puisse adopter une stratégie de description fonctionnelle identifiant des activités.

En synthèse, en service web décrit au moyen d'un langage de spécification de haut niveau — WSDL — des interfaces communiquant entre elles. Les données manipulées, aussi complexe que désirées, sont jointes à l'opération du service demandé. Elles peuvent être décrites dans un type structuré XML, binaires ou sérialisées. Les communications se réduisent à l'envoi d'une requête et à la reception d'une réponse de manière asynchrone. Les architectures de communications doivent être abstraites au moyen d'un langage de haut niveau — SOAP, par exemple — permettant la portabilité sur plusieurs médias. Il est possible d'adjoindre aux services, des services prédéfinis tels que ceux d'authentification, d'encryption, d'acquittement des données, de notification d'évènements, ...

1.2 WSDL

Le document [1] décrit au travers d'un manuel de référence le langage WSDL [1] 1.1. La caractérisation du comportement d'un service web [1.1] est clairement définie — ou le mieux possible —, leur utilisation est formalisée car, en effet, nous les utilisons depuis fort longtemps. Cependant, l'émergence des **N**ouvelles **T**echnologies de l'**I**nformation et de la **C**ommunication, de nouveaux langages, de nombreux *middelware*, ... font que leur utilisation ayant pour but l'implantation de services basés sur l'Internet deviennent de l'ordre de l'exploit. En effet, les techniques de génie logiciel assurant une certaine fiabilité, robustesse, rapidité d'implantation et de maintenance, portabilité, ... se verraient contrainte de laisser place à une grande part de *hack*. Ainsi le concept de *service web* a été formalisé et un langage de description de haut niveau lui a été associé.

WSDL est un langage au format XML décrivant les services web comme un ensemble de nœuds opérants sur des messages contenant soit des informations orientées document soit des informations orientées procédure. Les opérations et les messages sont décrits de manière abstraite, puis par la suite associés à un protocole réseau et un format de messages concrêts de sorte à définir un nœud de communication réel. Ces nœuds réels sont associés à un nœud abstrait: c'est un service. WSDL est extensible afin de permettre la description de nœuds et de messages sans se soucier du format des messages et des protocoles réseaux utilisés lors de la communication. Le document décrit uniquement l'utilisation des protocoles de communication cibles SOAP 1.1, HTTP get/post et MIME.

2 Spécifications de l'application

Motivation

Que signifie réaliser un projet? Réaliser un projet, tel que l'achat d'une maison par exemple, requiert plusieurs phases: prospection, étude de terrain, impôts locaux, accéssibilité au terrain, état de la maison, de toit, ... Lorsque ces étapes ont été franchies alors, il faut penser au financement: aller demander un crédit, prêts divers, notaire, multiples ententes, ... Puis finalement les papiers sont signer et la maison est achetée. Toutes ces démarches tendent à assurer une qualité de vie dans l'habitat. Bien entendu, cette qualité aurait été fortement compromise si l'achat de la maison

1. **W**eb **S**ervices **D**escription **L**anguage

avait directement suivi l'envie de l'acheter. Le pauvre propriétaire se serait rendu compte, trop tard, que le terrain est inondable, que le soleil se fait rare et que les thermites se régalent des charpentes.

La conception de logiciels subit les mêmes règles décrites par un *plan d'assurance qualité* et le respect d'un cycle de développement (cycle en V, cycle incrémental, extreme programming, ...). Des phases de test du code produit sont proposées dans ces cycles de sorte à valider les algorithmes, les spécifications puis le logiciel dans sa globalité à chaque étape du processus de développement.

Plusieurs outils d'aide au *test* existent, qui la plupart sont proposés dans des platte-formes complexes et onéreuses. Ainsi, l'application que je propose dans le but de mettre en œuvre une implantation de service web, propose à l'équipe de *test & intégration* d'assigner les erreurs trouvées dans l'application à un développeur afin qu'il le corrige. Cet outil destiné au *testing* quelqu'il soit — tests unitaires, tests d'intégration, tests de portabilité ou test fonctionnels — est tout indiqué pour améliorer la communication entre plusieurs personnes travaillant sur un même projet et la traçabilité des *bugs*.

2.1 Cas d'utilisations

Dans le cadre de l'ESSI, nous avons été amenés à réaliser des projets: c'est dans cette optique que l'outil sera susceptible de réagir de manière cohérente.

Cette section n'a pas pour but de présenter de manière exhaustive tous les cas d'utilisation possible dans le milieu industriel mais de proposer un sous-ensemble de cas permettant d'aboutir à quelques spécifications informelles.

Nous nous laissons néanmoins la possibilité d'extensions afin qu'il puisse être adapté à des besoins plus conséquents permettant en l'occurence le *passage à l'échelle*.

2.1.1 Tests unitaires

Nous admettrons que chaque programmeur teste régulièrement son code pendant les phases de développement.

T-TU-1 L'application pourra être utiliser de sorte à ce que chaque programmeur de l'équipe puisse enregistrer les éléments relatif au code à tester.

T-TU-2 Régulièrement ils pourront être avertis de l'état des tests à faire ou restant à faire.

T-TU-3 Lorsqu'un test à été réalisé, il change d'état. Le *chef de projet* pourra ainsi s'assurer de l'état d'avancement des codes individuels. Les programmeurs pourront visualiser et estimer le temps nécessaire à la clôture de leurs tâches avant la prochaine réunion, livraison, *milestone* ou autre.

T-TU-4 Les notes enrigistrées associées aux tests unitaires pourront être accompagnées d'exemples d'utilisation, de procédures de tests ou de cas réels de stimulation du morceau de code. Toutes ces notes pourront ainsi être réutilisée dans le cadre de tests de non-régression. Ici apparait un gain de temps lié aux probèmes de régression et à leur détection.

T-TU-5 Ces notes pourrons servir de base à la rédaction de documentations relatives aux API de communication entre les différentes parties du code.

2.1.2 Tests d'intégration

Nous admettrons que les différents codes produits par les programmeurs sont intégrés. Le but est de minimiser ce temps, cependant des erreurs peuvent apparaitre lors de la communication des données dans la structure générale du logiciel en cours de production. Ainsi, l'application est tout indiquée pour transmettre l'information aux programmeurs.

T-TI-1 Dès qu'une erreur est détectée, une note est envoyée à un programmeur afin qu'il la corrige.

T-TI-2 Chaque erreur est nécessairement accompagnée d'un exemple permettant de la reproduire. Il servira en outre au programmeur pour qu'il puisse vérifier qu'il a correctement corrigé l'erreur.

T-TI-3 A chaque note est associé un état qui précise

- l'erreur est assignée
- l'erreur est a été corrigée
- l'erreur n'en n'est pas une: effet de bord d'une autre erreur ou comportement incohérent devant être éliminer dans une phase amomt

T-TI-4 A chaque note est associé un status qui précise le niveau de priorité de correction de l'erreur

- sevère: l'erreur empêche la poursuite des tests
- grave : l'erreur engendre un comportement incohérent de l'application testée
- moyenne : l'erreur est ennuyeuse et provoque une gêne à l'utilisateur de l'application
- minimale: l'erreur n'a aucune incidence sur le système mais ne respecte pas une spécification

2.1.3 Tests fonctionnels

Nous admettrons que des tests fonctionnels relatifs aux spécifications émises par le client sont réalisé. Ces tests permettent de mettre en évidence des erreurs graves de conception ou de compréhension amenant à léchec d'un projet.

T-TF-1 Une erreur provenant d'un test fonctionnel devra être enregistrée afin d'être soumise au groupe en vue d'une discussion.

T-TF-2 Une telle erreur devra être accompagnée d'un exemple l'ayant provoquée et de la spécification violée.

T-TF-3 L'erreur porte un état précisant

- qu'elle doit être discutée
- qu'elle a été résolue par une nouvelle itération sur le code
- qu'elle pourra être résolue dans une prochaine version du produit
- qu'elle amène à l'échec du projet

2.1.4 Tests de portabilité

Nous admettrons que le code de l'application produite durant le projet a subi des tests de portabilité sur plusieurs systèmes d'exploitation, processeurs et compilateurs.

T-TP-1 Dès qu'une erreur est détectée, une note est envoyée à un programmeur afin que le code associé devienne portable.

T-TP-2 Chaque erreur est nécessairement accompagnée d'un exemple permettant de la reproduire.

T-TP-3 A chaque note est associé un état qui précise

- l'erreur est assignée
- l'erreur est a été corrigée
- l'erreur n'en n'est pas une: effet de bord d'une autre erreur ou comportement incohérent devant être éliminer dans une phase amomt

T-TI-4 A chaque note est associé un status qui précise le niveau de priorité de correction de l'erreur

- système : l'erreur est liée à un système d'exploitation particulier à préciser
- processeur : l'erreur est liée à un type de processeur particulier à préciser
- compilateur : l'erreur est liée à un compilateur particulier à préciser

2.1.5 Généralités

T-GEN-1 Tous les exemples joints aux notes pourront servir de base pour les test de non-régression.

T-GEN-2 Les notes pourront servir à mettre à jour une documentation pour la version testée précisant, pour la traçabilité de code les erreurs rencontrées. Certaines erreurs auront été corrigées alors que dáutres le seront dans une future version. Ici apparait un gain de temps d'un point de vu de la documentation et du nombre d'itérations sur le code estimées suffisantes pour atteindre la qualité désirée du produit.

2.2 Spécifications

Nous venons de déterminer au travers des spécifications informelles décrites ci-dessus, le cadre d'utilisation d'une telle application. Les cinq cas de tests suivants ont été exhibés:

i. tests unitaires
ii. tests d'intégration
iii. tests fonctionnels
iv. tests de protabilité
v. tests de non régression

Nous nous attachons, dans cette partie, à déterminer, en fonction des informations recueillies, d'une part les *services* requis et d'autre part les *données* à échanger.

2.2.1 Services

Nous ne nous attacherons pas ici à décrire les opérations de gestion des utilisateurs. Seules les opérations relatives à l'application elle-même seront décrites.

T-S-1 Quatre types de notes peuvent être envoyées sur le site ébergeant le gestionnaire de reports d'erreurs, relatives au type de test affilié (unitaire, d'intégration, fonctionnel ou de portabilité). Une opération de **soumission d'une note** est requis, nous l'appellerons **SubmitNote**.

T-S-2 Un utilisateur de l'application pourra consulter une liste de notes en les récupérant dans sa fenêtre de navigation web. Une opération de **récupération d'une liste de note**, appelé **GetNotes** devra donc être disponible.

T-S-3 Plusieurs exemples joints aux notifications de *bug* durant les phases de tests peuvent être utilisés comme programmes de tests de non-régression. Une opération de **récupération de programmes de tests de non-régression** appellée **GetTests** sera proposée.

T-S-4 Les notes rédigées tout au long du cycle de développement du logiciel peuvent constituer des éléments d'élaboration de documentations (traçabilité, versions, ...). Une opération de **récupération des informations relatives à la documentation** générée automatiquement devra permettre à l'utilisateur de suivre lévolution de *versions*. Nous appellerons cette fonctionnalité **GetDocumentation**.

2.2.2 Données

Les données transmises seront liées d'une part aux notes relatives au style de test soumis, et d'autre part aux requêtes formulées.

T-D-1 Toutes les notes soumises ont les champs communs suivants:

- **DateOfCreation**: date de notification
- **SubmitBy** : nom du créateur
- **Info** : ce champs enregistre toutes les modifications apportées à la note

T-D-3 Les notes de *tests unitaires*, de *tests d'intégration* et de *tests de portabilité* ont un état **SimpleState**. Un état peut prendre une et une seule des valeurs décrites ci-dessous:

- **assigned**: le *bug* est assigné à un programmeur en vu d'être corrigé. Il restera dans cet état jusqu'à ce que le programmeur le corrige et le fasse passer en **resolved**, ou qu'il détermine qu'il ne s'agit pas d'un *bug*. Dans ce cas là, l'état passera en **reviewed**
- **resolved**: le *bug* a été corrigé
- **reviewed**: le *bug* est conséquence d'un autre *bug*. Le comportement de l'application ayant levée cette erreur est cohérent

T-D-4 Les notes de *tests d'intégration* ont un status **SimpleStatus** qui permet au programmeur de décider de l'ordre de priorité de correction de l'erreur. Ce status peut prendre une et une seule des valeurs suivantes:

- **failure**: l'erreur empêche la poursuite des tests
- **error** : l'erreur engendre un comportement incohérent de l'application testée
- **warning**: l'erreur provoque une gêne dans l'utilisation de l'application testée
- **note** : l'erreur n'a aucune incidence. Le code l'ayant provoqué n'implante pas correctement un point de spécification

T-D-5 Les notes de *tests fonctionnels* ont un état **FunctionnalState** qui permettra au groupe d'adopter un comportement vis à vis de la viabilité de l'application en cours de construction. Cet état peut prendre une et une seule des valeurs suivantes:

- **submit**: le code associé ne respecte pas une spécification donnée par le client. Le groupe devra en discuter afin de trouver une solution. Le groupe décidera après délibération du changement d'état parmis ceux précisés ci-dessous
- **resolved**: le code peut être modifié sans répercution afin dímplanter la spécification
- **released**: la version du logiciel en cours d'implantation sera figée et livrée. La spécification sera implantée (ou le code modifié) lors d'une prochaine version
- **failure** : modifier le code ou implanter le spécification remet en cause l'architecture du logiciel et les choix faits. L'implanter reviendrait à démarrer un nouveau projet. Ainsi, le projet est déclaré en échec et arrêté de sorte à le reprendre depuis le début

T-D-6 Les notes de **tests de portabilité** ont un status **PortableStatus** précisant l'architecture ayant provoquée l'erreur. Ce status peut prendre une et une seule des valeurs suivantes:

- **system** : l'erreur est liée au système d'exploitation sous lequel l'application est testée: Linux, Windows95, WindowsNT, SunOs, ...
- **processor**: l'erreur est liée au processeur pour lequel l'application est testée: Intel, Sparc, ...
- **compiler** : l'erreur est liée au compilateur avec lequel l'application est compilée: gcc, cl.exe, jikes, javac, ...

T-D-7 Les *tests d'intégration* et les *tests de portabilité* demandent d'être assignés à un programmeur. Ce champs sera nommé **AssignedTo**.

T-D-8 Les *tests fonctionnels* sont adressés, dès que de telles erreurs sont notifiées, à tous les programmeurs de groupe.

T-D-9 Les *tests unitaires* sont adressés au programmeur l'ayant proposé.

T-D-10 Les *tests d'intégration*, *tests fonctionnels* et *tests de portabilité* peuvent faire part de la construction d'une documentation de version. Ainsi, la valeur logique **Doc** devra préciser:

- **false**: le test ne participera pas à l'élaboration de la documentation
- **true** : le test participera à l'élaboration de la documentation. Danc ce cas, une information complémentaire **DocText** devra être communiquée. Le testeur ajoutera par ce biais une note personnelle

T-D-11 Les *tests unitaires, tests d'intégration, tests fonctionnels* et *tests de portabilité* peuvent faire part de la construction d'une base de tests de non-régression. Ainsi, la valeur logique **Test** devra préciser:

- **false**: le test ne participera pas à l'élaboration de la base
- **true** : le test participera à l'élaboration de la base. Danc ce cas, une information complémentaire **DocText** devra être communiquée. Le testeur ajoutera par ce biais une note personnelle

T-D-12 Les exemples permettant de reproduire les erreurs doivent être fournis pous les *tests d'intégration, tests fonctionnels* et *tests de portabilité*. Ces données seront des fichiers sources joints lors de la requête de soumission.

T-D-13 Les *tests fonctionnels* doivent préciser dans le cas où une telle erreur soit notifiée la spécification violée **Spec** en précisant:

- **doc** : le document relatifà la spécification
- **release**: la version du document
- **spec** : le numéro ou l'identificateur de la spécification

T-D-14 Les programmeurs sont enregistrés dans la base, ils sont ainsi nominativement déterminés.

T-D-15 Un programmeur peut récupérer les notes qui lui sont assignées. Une liste de ces notes — au format du *test* dans lequel elles auront été décrites — lui est envoyée.

T-D-16 Un programmeur peut récupérer les tests unitaires constituant la base. Il récupère alors une liste de fichiers source.

T-D-17 Les utilisateurs peuvent récupérer des informations (notes, fichiers source, documentation) provenant de la base en précisant des critères de recherches. Ces requêtes seront encapsulées dans **Request**.

2.2.3 Schéma

Les sections précédentes précisent en fonction des scénarii de cas d'utilisateur les opérations requises et le format des données supportées. Des identificateurs sont associés aux opérations et aux données de sorte de pouvoir les situer dans un contexte de manière non ambigüe: durant la phase d'implantation par exemple.

Nous traitons ici des seuls aspects liés aux services requis et non aux systèmes de gestion des interfaces graphiques permettant la visualisation des données, ni à la base de données gérant les notes d'erreurs, ni aux requêtes que pourraient formuler les utilisateurs, ... représentés par le schéma 1.

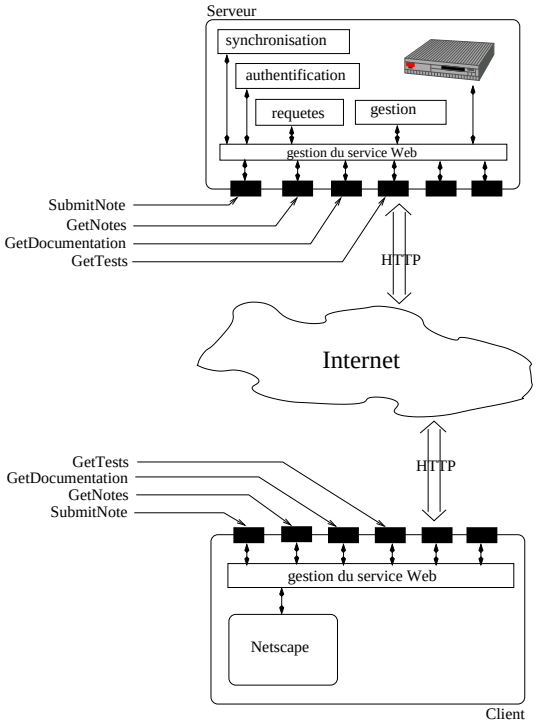

FIG. 1 –. Application

2.3 Services Web

Opérations et données étant maintenant définis, associons les ensembles.

2.3.1 Données structurées

Les *requirements* formulés ci-dessus sur les données va nous permettre délaborer des types de données complexes permettant la construction des différentes notes.

T-TU-DATA Les notes de *tests unitaires* sont définis par les données ci-dessous:

- **DateOfCreation** (T-D-1): date de notification de la note
- **SubmitBy** (T-D-2, T-D-14): nom d'enregistrement de l'utilisateur qui notifie la note
- **Info** (T-D-3): trace des opérations réalisées sur la note
- **SimpleState** $\in \{assigned, resolved, reviewed\}$ (T-D-3): état associé à la note

Cet ensemble de données sera nommé **DataUserTest**.

T-TI-DATA Les notes de *tests d'intégration* sont définis par les données ci-dessous:

- **DateOfCreation** (T-D-1): date de notification de la note
- **SubmitBy** (T-D-2, T-D-14): nom d'enregistrement de l'utilisateur qui notifie la note
- **Info** (T-D-3): trace des opérations réalisées sur la note
- **SimpleState** $\in \{assigned, resolved, reviewed\}$ (T-D-3): état associé à la note
- **SimpleStatus** $\in \{failure, error, warning, note\}$ (T-D-4): status associé à la note
- **AssignedTo** (T-D-7, T-D-14): nom d'enregistrement de l'utilisateur à qui la note est destinée
- **Doc** $\in \{true, false\}$ (T-D-10): ce champs précise si cette note devra être ou pas mentionnée dans la documentation
- **DocText** (T-D-10): dans le cas où le champs **Doc** soit positionné à **true** alors l'utilisateur pourra ajouter un texte associé à la note pour être inclus dans la documentation
- **Files** (D-12): des fichiers au format texte sont joints à la note

Cet ensemble de données sera nommé **DataIntegrationTest**.

T-TF-DATA Les notes de *tests fonctionnels* sont définis par les données ci-dessous:

- **DateOfCreation** (T-D-1): date de notification de la note
- **SubmitBy** (T-D-2, T-D-14): nom d'enregistrement de l'utilisateur qui notifie la note
- **Info** (T-D-3): trace des opérations réalisées sur la note
- **FunctionnalState** $\in \{submit, resolved, released, failure\}$ (T-D-5): état associé à la note
- **Doc** $\in \{true, false\}$ (T-D-10): ce champs précise si cette note devra être ou pas mentionnée dans la documentation
- **DocText** (T-D-10): dans le cas où le champs **Doc** soit positionné à **true** alors l'utilisateur pourra ajouter un texte associé à la note pour être inclus dans la documentation
- **Files** (T-D-12): des fichiers au format texte sont joints à la note
- **Spec** $\triangleq$ $(doc, release, spec)$ (T-D-13): spécification violée

Cet ensemble de données sera nommé **DataFunctionnalTest**.

T-TP-DATA Les notes de *tests de portabilité* sont définis par les données ci-dessous:

- **DateOfCreation** (T-D-1): date de notification de la note
- **SubmitBy** (T-D-2, T-D-14): nom d'enregistrement de l'utilisateur qui notifie la note
- **Info** (T-D-3): trace des opérations réalisées sur la note
- **SimpleState** $\in \{assigned, resolved, reviewed\}$ (T-D-3): état associé à la note
- **PortableStatus** $\in \{system, processor, compiler\}$ (T-D-6): status associé à la note
- **AssignedTo** (T-D-7, T-D-14): nom d'enregistrement de l'utilisateur à qui la note est destinée
- **Doc** $\in \{true, false\}$ (T-D-10): ce champs précise si cette note devra être ou pas mentionnée dans la documentation
- **DocText** (T-D-10): dans le cas où le champs **Doc** soit positionné à **true** alors l'utilisateur pourra ajouter un texte associé à la note pour être inclus dans la documentation
- **Files** (T-D-12): des fichiers au format texte sont joints à la note

Cet ensemble de données sera nommé **DataPortabilityTest**.

2.3.2 Opérations et données

Cette partie est consacrée à associer les opérations avec les données déterminées ci-dessus.

T-LINK-1 L'opération de soumission d'une note **SubmitNote** (T-S-1) permet d'enregistrer une note associée à un des quatre types de tests définis (T-TU-1, T-TI-1, T-TF-1, T-TP-1). Il doit ainsi être possible d'associer à cette opération des données d'un des types représentant ces tests (T-TU-DATA, T-TI-DATA, T-TF-DATA, T-TP-DATA).

T-LINK-2 L'opération de récupération de notes **GetNotes** (T-S-2) permet de récupérer une liste de notes en fonction de critères définis par l'utilisateur. Pour les besoins de l'application nous réduirons ces requêtes à récupérer les notes assignées à l'utilisateur — enegistré dans la base — qui émet la requête (T-D-17).

T-LINK-3 L'opération de récupération de fichiers source **GetTests** (T-S-3) permet de récupérer une liste de fichiers source en fonction de critères définis par l'utilisateur (T-D-17).

T-LINK-4 L'opération de récupération de la documentation **GetDocumentation** (T-S-4) permet de récupérer la documentation créée en fonction de critères définis par l'utilisateur (T-D-17).

3 Spécifications WSDL de l'application

Le document [3, introduction] présente les sept points permettant de réaliser une spécification WSDL.

i. **Types**: environnement pour les définitions des types utilisateur pouvant utiliser des types primitifs
ii. **Message**: définitions abstraites des données typées qui seront communiquées
iii. **Operation**: descriptions abstraites d'une action supportée par le service
iv. **Port Type**: ensembles abstraits d'opérations supportées par un ou plusieurs nœuds de communication
v. **Binding**: protocoles concrets et spécifications du format des données pour un type de port particulier
vi. **Port**: nœuds de communication définis en tant qu'associations de liens et d'adresses réseau
vii. **Service**: un ensemble de nœuds de communication

3.1 Section Types

3.1.1 Eléments

Cette section est consacrée à la définition des éléments utilisés.

T-XML-ELEMENT-1 Définissons le champs **DateOfCreation**.

```
<xsd:element
  name="DateOfCreation"
  id="DateOfCreation"
  substitutionGroup=\
    "DateOfCreationAbstractType"
  minOccurs=''1''
  maxOccurs=''1''
>
  <xsd:annotation>
   <xsd:documentation
     xml:lang=''en''
   >
     Date of note notification
   </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

T-XML-ELEMENT-2 Définissons le champs **SubmitBy**.

```
<xsd:element
  name="SubmitBy"
  id="SubmitBy"
  substitutionGroup=\
    "SubmitByAbstractType"
  minOccurs=''1''
  maxOccurs=''1''
>
  <xsd:annotation>
   <xsd:documentation
     xml:lang=''en''
   >
     User who submit the note
   </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

T-XML-ELEMENT-3 Définissons le champs **Info**.

```
<xsd:element
  name="Info"
  id="Info"
  substitutionGroup="InfoAbstractType"
  minOccurs=''1''
  maxOccurs=''unbound''
>
  <xsd:annotation>
   <xsd:documentation
     xml:lang=''en''
   >
     Information to trace the note
   </xsd:documentation>
  </xsd:annotation>
</xsd:element>
```

T-XML-ELEMENT-4 Les divers champs spécifiés en T-D sont déclarés en **element** de type abstrait permettant de leur associer, lors de l'instantiation, un type dérivé le plus adapté à son utilisation. En effet, lors d'une phase de *debug* ou d'*analyse fonctionnelle* de l'application mettant en œuvre les services, le type dérivé pourra être très simple. D'un autre côté, dans l'optique d'un déploiement, le type dérivé pourra être beaucoup mieux précisé. Le but est évidamment de minimiser les modifications à apporter au code.

Les éléments seront écrit de la même façon que ceux présentés en T-XML-ELEMENT-1..4.

3.1.2 Types

Cette section est consacrée à la définition des types utilisés.

T-XML-TYPE-1 Définissons le type **DateOfCreation-Type**.

```
  <xsd:complexType
    name=''DateOfCreationAbstractType''
    abstract=''true''
  >
    <xsd:annotation>
     <xsd:documentation
       xml:lang=''en''
     >
Abstract type denoted the date of creation
     </xsd:documentation>
    </xsd:annotation>
  </xsd:complexType>

  <xsd:complexType
    name=''DateOfCreationType''
  >
    <xsd:annotation>
     <xsd:documentation
       xml:lang=''en''
     >
       Type denoted the date of creation
     </xsd:documentation>
    </xsd:annotation>

    <xsd:complexContent>
      <xsd:extension base=\
   ''DateOfCreationAbstractType''>
        <xsd:attribute
         name=''date''
         type=''xsd:date''
         use=''required''
      </xsd:attribute>
    </xsd:complexContent>
  </xsd:complexType>
```

T-XML-TYPE-2 Le type de la note soumise est déclaré comme suit:

```
  <complexType
    name=''DataTest''
    id=''DataTest''
  >
    <choice>
      <element
        name=''DataUserTest''
        type=''DataUserTestType''
      >
        <annotation>
          <documentation
            xml:lang=''en''
          >
            Data for user test
          </documentation>
        </annotation>
      </element>

      <element
        name=''DataIntegrationTest''
        type=''DataIntegrationTestType''
      >
        <annotation>
          <documentation
            xml:lang=''en''
          >
            Data for integration test
          </documentation>
        </annotation>
      </element>

      <element
        name=''DataFunctionnalTest''
        type=''DataFunctionnalTestType''
      >
        <annotation>
          <documentation
            xml:lang=''en''
          >
            Data for functionnal test
          </documentation>
        </annotation>
      </element>

      <element
        name=''DataPortabilityTest''
        type=''DataPortabilityTestType''
      >
        <annotation>
          <documentation
            xml:lang=''en''
          >
            Data for portability test
          </documentation>
        </annotation>
      </element>
    </choice>
  </complexType>
```

T-XML-TYPE-3 Les autres types nécessaires à l'application seront décrits de la même façon qu'en T-XML-TYPE-2. Le type abstrait associé à un élément est décrit. Ensuite, un type dérivé est spécifié de sorte à ce qu'il représente fidèlement la sémantique de l'élément au quel il se rapporte.

3.1.3 Déclaration de types WSDL

La section <**types**> contiendra Les éléments définissant les données qui transiteront sur le réseau:

- **SubmitNoteRequest** (T-LINK-1): cette donnée représentera le type de la note communiquée
- **GetNotesRequest** (T-LINK-2): cette donnée représentera la requête demandant la liste de notes
- **GetTestsRequest** (T-LINK-3): cette donnée représentera la requête demandant la liste des fichiers sources
- **GetDocumentationRequest** (T-LINK-4): cette donnée représentera la requête demandant la documentation
- **GetNotesAnswer** (T-LINK-2, T-XML-ELEMENT-4, T-XML-TYPE-3): cette donnée représentera la liste de notes retournées par le serveur lorsqu'un client en fait la demande par **GetNotesRequest**
- **GetTestsAnswer** (T-LINK-3, T-XML-ELEMENT-4, T-XML-TYPE-3): cette donnée représentera la liste de notes retournées par le serveur lorsqu'un client en fait la demande par **GetTestsRequest**
- **GetDocumentationAnswer** (T-LINK-4, T-XML-ELEMENT-4, T-XML-TYPE-3): cette donnée représentera la liste de notes retournées par le serveur lorsqu'un client en fait la demande par **GetDocumentationRequest**

SubmitNoteRequest

```
<element
  name=''SubmitNoteRequest''
  id=''SubmitNoteRequest''
  type=''DataType''
>
  <annotation>
    <documentation
      xml:lang=''en''
    >
      Data denoted a note
    </documentation>
  </annotation>
</element>
```

GetNotesRequest

```
<element name=''GetNotesRequest>
  <complexType>
    <sequence
      maxOccurs=''unbound''
    >
      <element
        name=''Data''
        type=''DataType''
      >
        <annotation>
          <documentation
            xml:lang=''en''
          >
            Data denoted a set of notes
          </documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>
</element>
```

GetTestsRequest est défini de façon similaire à GetNotesRequest.

GetDocumentationRequest est défini de façon similaire à GetNotesRequest. La documentation, étant du texte, pourra être structurée dans un format XML-like de sorte à pouvoir y appliquer des styles d'affichage différents (HTML, RTF, PDF, ...) fonction des utilisations requises.

3.2 Section Message

La section <**Message**> est consacrée aux définitions abstraites des données typées qui seront communiquées. Les messages sont:

- **SubmitNoteDataInput** (T-LINK-1): donnée portant la soumission d'une note
- **GetNotesDataInput** (T-LINK-2): donnée portant la requête de récupération des notes
- **GetTestsDataInput** (T-LINK-3): donnée portant la requête de récupération des fichiers source
- **GetDocumentationDataInput** (T-LINK-4): donnée portant la requête de récupération de la documentation
- **GetNotesDataOutput** (T-LINK-2): donnée portant la récupération des notes
- **GetTestsDataOutput** (T-LINK-3): donnée portant la récupération des fichiers source
- **GetDocumentationDataOutput** (T-LINK-4): donnée portant la récupération de la documentation

Les *messages* suffixé par **Input** seront de la forme suivante:

```
<message name=''SubmitNoteDataInput''>
  <part
    name=''body''
    element=''xsd1:SubmitNoteRequest''
  />
</message>
```

Ceux suffixé par **Output** seront de la forme analogue suivante:

```
<message name=''GetNotesDataOutput''>
  <part
```

```
    name=''body''
    element=''xsd1:GetNotesAnswer''
  />
</message>
```

3.3 Section Port Type

La section <**Port Type**> définit des ensembles abstraits d'opérations supportées par un ou plusieurs nœuds de communication. Des opérations définissant les services désirés utilisent les messages définis ci-dessus. Pour chacun d'eux, un port leur est associé:

- **SubmitNotePortType** (T-S-1): déclare un port dont la seule donnée **SubmitNoteDataInput** sera sortante
- **GetNotesPortType** (T-S-2): déclare un port dont la donnée **GetNotesDataInput** sera émise et **GetNotesDataOutput** attendu
- **GetTestsPortType** (T-S-3): déclare un port dont la donnée **GetTestsDataInput** sera émise et **GetTestsDataOutput** attendu
- **GetDocumentationPortType** (T-S-4): déclare un port dont la donnée **GetDocumentationDataInput** sera émise et **GetDocumentationDataOutput** attendu

Par exemple, **GetTestsPortType** sera de la forme:

```
<portType name=''GetTestsPortType''>
  <operation name=''GetTests''>
    <input message=\
 ''tns:GetTestsDataInput''/>
    <output message=\
 ''tns:GetTestsDataOutput''/>
  </operation>
</portType>
```

3.4 Section Binding

La section <**Binding**> associe aux ports, aux opérations et aux messages un protocole de communication. Nous choisirons le protocole SOAP sur HTTP. Pour chaque élément défini dans la section **Port Type**, un **Bind** le liera avec les couches logicielles de plus bas niveau.

- **SubmitNoteBinding** liera **SubmitNotePortType**
- **GetNotesBinding** liera **GetNotesPortType**
- **GetTestsBinding** liera **GetTestsPortType**
- **GetDocumentationBinding** liera **GetDocumentationPortType**

L'implantation de **GetTestsBinding** pourra être de la forme suivante:

```
<binding name=''GetTestsBinding''
    type=''tns:GetTestsPortType''
  >
  <soap:binding
    style=''document''
    transport=''http://schemas.\
      xmlsoap.org/soap/http"
  />
  <operation name="GetTests">
    <soap:operation
      soapAction="http://\
        my_web_site/GetTests"
    />
    <input>
      <soap:body
        use="literal"
      />
    </input>
    <output>
      <soap:body
        use="literal"
      />
    </output>
  </operation>
</binding>
```

3.5 Section Service

La section <**Service**> permet de déclarer une interface publiant les services offerts par l'application de gestion des *bugs*:

- **SubmitNoteService** pour le service de soumission d'une notification de *bug*
- **GetNotesService** pour le service de récupération de notes
- **GetTestsService** pour le service de récupération de fichiers source
- **GetDocumentationService** pour le service de récupération de la documentation

Le service **GetTestsService** pourrait être décrit de la façon suivante:

```
<service name=''GetTestsService''>
  <documentation>
    The service returns files
  </documentation>
  <port
    name=''GetTestsPort''
    binding=''tns:GetTestsBinding''
  >
    <soap:address
      location=''http://my_web_site/\
```

```
                gettests''
    />
  </port>
</service>
```

Conclusion - Perspectives

Les spécifications décrites ci-dessus vont permettre de passer à l'étape suivante: spécifications d'implantation et codage. Le langage cible devra être en l'occurence choisi de sorte à pouvoir exprimer le potentiel requis par une telle application.

D'autres méthodes mettant en œuvre l'implantation de services web de manière plus simple existent comme par exemple celle proposée par la plate-forme .NET de Microsoft© en déclarant des *web methods*.

Références

[1] Web Services Description Language 1.1, Ariba, International Business Machines Corporation, Microsoft.

[2] Web Service Use Cases Scenarios, www.w3.org/2002/ws/desc

[3] A web service Primer, Venu Vasudevan, www.xml.com

[4] Dr. Tom's Guide to XSD, IMS Global Learning Consortium, Inc, 2001

[5] XML Schemas, http://www.oasis-open.org/cover/schemas.html