

Comprendre le besoin et vérifier une exigence

Analyse : de l'exigence à la spécification

Ressource R3.03

BUT Informatique, début de deuxième année

Version du 24 septembre 2026

À la fin de ce cours

Vous saurez distinguer un besoin d'une solution, écrire une exigence vérifiable et relier cette exigence à une observation.

- Notre exemple de cours : les emprunts d'une bibliothèque.
- En TD, vous appliquerez la même démarche à un autre dossier.
- En C, vous apprendrez à retrouver le rôle d'un fichier et à suivre un résultat.

#	Partie	Durée
1	Situation de départ	10 min
2	Besoin, exigence et solution	20 min
3	Vérification d'une exigence	15 min
4	Exercice et correction	15 min
5	Lecture C : structures et fichiers	10 min
6	Synthèse et passage au TD	5 min

Cas fictif utilisé dans les quatre cours

« Il nous faudrait un bouton qui envoie un rappel tous les jours aux lecteurs en retard. »

Travail individuel, deux minutes : quelle question poser avant de parler de bouton ou de courriel ?

Cas fictif utilisé dans les quatre cours

« Il nous faudrait un bouton qui envoie un rappel tous les jours aux lecteurs en retard. »

Travail individuel, deux minutes : quelle question poser avant de parler de bouton ou de courriel ?

Une entrée possible dans l'entretien

« Racontez-moi la dernière situation où un retour tardif a posé problème. »

L'entretien fait apparaître un problème concret

Information recueillie	Ce qu'elle permet de comprendre
Des lecteurs attendent un exemplaire qui aurait dû revenir. Les rappels manuels prennent du temps chaque matin. Un rappel identique part parfois après le retour du livre.	Le retard pénalise aussi d'autres lecteurs. Le personnel a une tâche répétitive. Le suivi du retour et l'envoi doivent être cohérents.
Ces faits appartiennent à notre cas fictif. Ils ne démontrent pas encore que l'envoi quotidien est la meilleure réponse.	

Demande « Ajouter un bouton de rappel quotidien. » Une solution envisagée.

Besoin « Réduire l'attente des lecteurs et le travail manuel de suivi. » Un problème et une valeur attendue.

Exigence « Un emprunt enregistré comme rendu ne doit plus figurer dans les rappels à préparer. » Une propriété du système.

Question de contrôle

Peut-on proposer une autre solution qui satisfait le même besoin ? Si oui, on a commencé à séparer besoin et solution.

Une question ouvre ou ferme l'enquête

Question qui suggère une réponse

Vous voulez un courriel quotidien ?

Il faut un bouton rouge ?

Le système doit être rapide ?

Question qui recherche un fait

Comment les lecteurs découvrent-ils leur retard aujourd'hui ?

Comment repérez-vous les situations urgentes ?

À quel moment une attente vous empêche-t-elle de travailler ?

Une question fermée sert ensuite à confirmer une règle précise. Elle devient gênante lorsqu'elle remplace l'exploration du besoin.

Un fait et une hypothèse ont des statuts différents

Statut	Exemple	Action
Fait du cas	Un rappel est parfois préparé après le retour.	Conserver sa source.
Hypothèse	Un rappel quotidien suffirait.	Demander confirmation.
Objectif proposé	Une liste affichée en moins de deux secondes.	Définir le contexte et faire valider le seuil.
Choix technique	Stocker les emprunts dans un fichier.	Justifier après l'analyse.
Un chiffre précis inventé reste une hypothèse.		

EF : que doit permettre ou faire le système ?

« La bibliothécaire peut enregistrer le retour d'un exemplaire. »

- Acteur : la bibliothécaire.
- Action : enregistrer un retour.
- Résultat observable : l'emprunt devient clos et l'exemplaire disponible.

Une règle complète ce service : un emprunt déjà clos ne peut pas être clos une seconde fois.

Dimension	Question à poser
Performance	Sur quel volume et quelle machine mesure-t-on le délai ?
Fiabilité	Après quel incident, quelle information doit rester correcte ?
Utilisabilité	Quelle tâche doit réussir quel utilisateur, avec quelle aide ?
Sécurité	Qui peut consulter ou modifier quelle information ?
Maintenabilité	Quelle évolution doit rester localisée et vérifiable ?

Ces catégories aident à chercher des exigences. Elles ne dispensent jamais d'écrire un critère de vérification.

- « Le système permet l'export de la liste des retards » : fonctionnalité.
- « Cet export conserve les accents et les dates » : propriété de qualité du résultat, à préciser.
- « L'export utilise la bibliothèque X » : contrainte technique, si elle est imposée et justifiée.

Ce qu'on évalue

La justification de la catégorie et la possibilité de vérifier la phrase comptent davantage qu'un classement sans explication.

Formulation trop vague

« La liste des retards doit s'afficher rapidement. »

Objectif proposé, à faire valider

Sur le poste de référence et un jeu de 10 000 emprunts, au moins 95 des 100 ouvertures de la liste se terminent en moins de deux secondes, après le chargement initial.

La population, la machine, le début et la fin de la mesure, le seuil et les répétitions sont identifiables. Le protocole reste à préciser.

Exigence

Un emprunt clos ne doit plus apparaître dans la liste des retards.

- ❶ Préparer un emprunt en retard et un emprunt clos.
- ❷ Demander la liste des retards.
- ❸ Vérifier la présence du premier et l'absence du second.

Tester seulement que la liste n'est pas vide ne vérifie pas cette exigence.

Recherche individuelle puis échange, six minutes.

- ① Le logiciel doit être fiable.
- ② La bibliothécaire peut enregistrer un retour.
- ③ Les emprunts sont stockés dans une liste chaînée.
- ④ Après un refus de renouvellement, l'échéance reste inchangée.

Pour chaque phrase : catégorie, information manquante et observation qui permettrait de décider si elle est satisfaite.

Phrase	Analyse possible
1	Qualité trop vague. Préciser un incident et une garantie après cet incident.
2	Fonctionnelle. Il manque le résultat et les préconditions de l'enregistrement.
3	Choix de conception. Demander quelle contrainte le justifie.
4	Règle de comportement observable. Comparer l'échéance avant et après le refus.

Pour la phrase 4, accepter plusieurs classements argumentés. Exiger dans tous les cas un test qui pourrait échouer.

Source	Un lecteur reçoit un rappel après avoir rendu le livre.
Exigence B-01	Un emprunt clos est absent de la liste des retards.
Décision à lire	La fonction qui sélectionne les emprunts à relancer.
Test T-01	Un emprunt en retard, un clos. Seul le premier apparaît.

Cette ligne permet de demander : si la règle change, quelles parties du logiciel et quels tests faut-il revoir ?

Une structure regroupe des données nommées

```
/* bilan.h */
#ifndef BILAN_H
#define BILAN_H
typedef struct {
    int emprunts_examines;
    int retards;
} bilan_t;
int aucun_retard(const bilan_t *b);
#endif
```

bilan_t nomme un type. Les deux champs appartiennent à chaque objet de ce type. La dernière ligne déclare une fonction.

Déclaration, définition et appel sont distincts

```
/* bilan.c */
#include "bilan.h"
int aucun_retard(const bilan_t *b) {
    return b->retards == 0;
}
/* main.c : extrait d'une autre unite de compilation */
bilan_t bilan = { .emprunts_examines = 7, .retards = 2 };
int resultat = aucun_retard(&bilan);
```

`&bilan` est l'adresse de l'objet. `b->retards` lit son champ à travers cette adresse. Ici, `resultat` vaut 0.

<code>bilan.h</code>	Déclare le type et le contrat partagé. Les gardes évitent une répétition dans une même unité de compilation.
<code>bilan.c</code>	Définit la fonction. Il inclut son en-tête pour vérifier la cohérence de la signature.
<code>main.c</code>	Appelle la fonction. Il inclut la déclaration, sans recopier sa définition.
<code>Makefile</code>	Décrit comment compiler puis réunir les fichiers objets.

Inclure un en-tête ne compile pas automatiquement le fichier `.c` correspondant.

Exercice de trace : prédire avant d'exécuter

```
bilan_t a = { .emprunts_examines = 7, .retards = 2 };  
bilan_t copie = a;  
copie.retards = 0;  
printf("%d %d\n", aucun_retard(&a), aucun_retard(&copie));
```

Deux minutes : que s'affiche-t-il ? Pourquoi modifier `copie.retards` ne modifie-t-il pas `a.retards` ?

```
bilan_t a = { .emprunts_examines = 7, .retards = 2 };  
bilan_t copie = a;  
copie.retards = 0;  
printf("%d %d\n", aucun_retard(&a), aucun_retard(&copie));
```

Deux minutes : que s'affiche-t-il ? Pourquoi modifier `copie.retards` ne modifie-t-il pas `a.retards` ? **Correction** : 0 1. L'affectation entre structures copie ici les deux entiers. Les deux objets ont des adresses distinctes.

Compilation Un champ inexistant ou un type incompatible. Le compilateur analyse chaque unité.

Édition des liens Une fonction déclarée et appelée n'a aucune définition dans les objets réunis.

Exécution Le programme compile mais prend une mauvaise décision. Il faut un test du comportement.

Habitude de travail

Lire le premier diagnostic, retrouver la déclaration, corriger puis recompiler. Une compilation réussie ne démontre pas une exigence.

- ❶ Partir d'un fait et en conserver la source.
- ❷ Distinguer besoin, exigence et choix de solution.
- ❸ Décrire le résultat observable et les conditions de vérification.
- ❹ Retrouver ensuite la responsabilité concernée dans le code.

Auto-évaluation : pouvez-vous expliquer un champ de structure, retrouver un prototype et proposer un contre-exemple à une exigence vague ?

Votre dossier : BioSeq

Au TD1, vous construirez cinq exigences prioritaires à partir du dossier client. Vous distinguerez les faits des hypothèses et proposerez une observation pour chacune.

Le TP utilisera votre exigence sur le résultat d'une sauvegarde

Vous lirez un programme fourni, suivrez les champs d'un rapport et corrigerez un bilan ambigu. Le parcours des répertoires est déjà écrit.