Nantes Université

BUT Informatique - 2024-2025 / Automates et Langages
(Christian Attiogbé)

Compléments pour la SAE 4.1

Applications Clients-Serveur : modélisation

Contexte. Dans le cadre de votre SAE 4.1, vous devez, (cf R4.01 - Architecture logicielle), *. . . créer plusieurs micro-services communicants, facilement déployables sur les VM de l'IUT. A minima devront être créés 3 micro-services, parmi lesquels,* ***un service reposant sur un jeu de données externes de votre choix***.

Nous pouvons décrire une application à base de micro-services communicants, comme une application avec une architecture de type **Clients-Serveur**. C'est ce que vous allez faire ici, pour le (micro-)service reposant sur le jeu de donné que vous avez choisi.

Les attentes de la SAE par rapport à l'enseignement "Automates et Langages (R4.12)" sont comme suit :
En amont de la programmation des applications, les phases d'analyse et de modélisation sont capitales pour la réussite, la qualité et la maintenance des applications.

Vous allez vous appuyer sur l'analyse et la modélisation à l'aide des Automates à états. Par exemple, le comportement d'un (micro-)service est représenté par un automate. Une trace du comportement/automate du service représente ainsi le scenario d'exécution du service. En amont de votre application, à l'issue de la phase d'analyse, vous devez représenter chaque service principal (ou des interactions entre services) par un automate ; attention à la granularité des (micro-)services, pour adapter votre modélisation. En phase de programmation, vous devez vous assurer que le comportement des (micro-)services corresponde à leurs modèles sous forme d'automates ; par exemple, les traces de vos automates fournissent systématiquement les scénario de tests que vous allez faire. Vous pouvez aussi extraire les squelettes de codes/programmes à partir de vos automates. Vos modèles (automates à états) constituent aussi une documentation de maintenance pour de futurs

développeurs qui vont lire, comprendre et éventuellement améliorer ou migrer votre application vers d'autres langages/technos/plateformes.

Le livrable attendu est un document synthétique (en format pdf, explicitement identifié avec une page de garde (le « qui-quoi-quand-où »)

- bien structuré (introduction, développement du corps, conclusion),
- bien argumenté (les choix/arguments sur le fond, les illustrations),
- bien écrit (structure, orthographe, grammaire, règles de typographie française), rendant compte non seulement du travail de [analyse et]modélisation que vous avez effectué, mais aussi, le cas échéant, comment cela vous a aidé pour la programmation et les tests, etc ; les apprentissages critiques et ou compétences que ce travail vous aura apporté, et tout autre élément mettant en valeur votre travail.

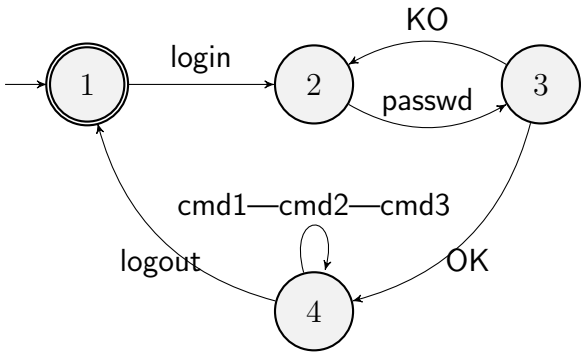
Des consignes supplémentaires seront données à la demande ou sur le forum de la page MADOC de la SAE.

Voici quelques pistes qui peuvent vous inspirer : `https://developers.redhat.com/articles/2021/11/23/how-design-state-machines-microservices#what_is_a_state_machine_`, `https://levelup.gitconnected.com/modelling-saga-as-a-state-machine-cec381acc3ef`

`https://www.quora.com/Can-we-create-Microservices-as-State-Machines`, `https://medium.com/@pradeep_thomas/business-flows-and-state-machines-part-ii-56fdd8414268`

Nous donnons ici quelques éléments pouvant vous aider à la modélisation de votre application.

Exemple vu dans le cours Modèle à états d'un processus client qui se connecte à un processus serveur (pour une interaction/coopération à la réalisation d'une tâche).



A partir de cet exemple, et des autres exercices vus en TD, vous pouvez modélisation une bonne partie de votre application Client-Serveur dans le cadre de la SAE. Le serveur étant basé sur des micro-services, son comportement est beaucoup plus simple que celui du client.

Hypothèses pour la modélisation

Nous utilisons ici les hypothèses suivantes :

— on extrait l'alphabet des clients et serveur, en faisant l'abstraction de leur comportement ;
— clients et serveur partagent une partie de l'alphabet ;
— les actions communes de l'alphabet, faites par le client et le serveur, sont synchronisées ;
— pour expliciter le sens de la communication, l'émetteur d'une action `alpha` utilisera `!alpha`, alors que le récepteur utilisera `?alpha`.
Dans ce contexte, le caractère `?` représente l'attente alors que le caractère `!` représente l'émission, et il y a synchronisation entre les deux ;
— on peut utiliser des paramètres à une action ; par exemple `beta(prm)`
— on peut utiliser, dans le cas asynchrone, les primitives `send(...)` pour émettre et `read(...)` pour recevoir ou lire.

Modélisation d'une application Client-serveur

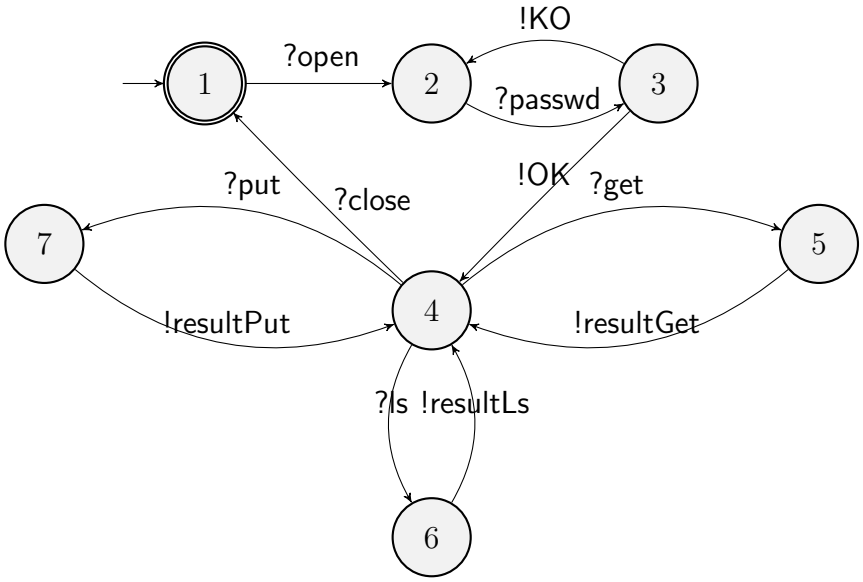
Modélisons un client-serveur FTP : le protocole de transfert de fichier sur le réseau. Dans ce protocole, un serveur est lancé au préalable et attend des connexions des clients. Un client tente de se connecter au serveur avec la commande `close`, après la phase de connexion, le client peut effectuer des commandes (`get, put, ls, ..., close`). La commande `close` permet au client d'arrêter l'interaction.

Nous n'avons pas envisagé l'hypothèse de serveurs multiples avec les clients multiples.

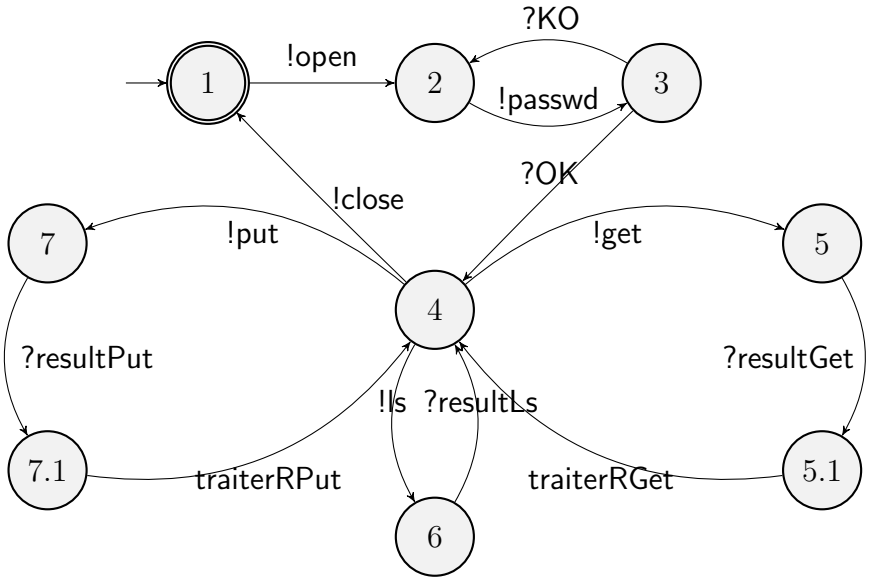
Modélisation du serveur Le serveur attend un client, puis interagit avec lui jusqu'à l'action `close`, et on recommence.

Ce serveur est élémentaire, sans traitement d'erreur, sans utilisation de paramètres, etc. Ainsi vous pouvez librement vous en inspirer.

Nous n'avons pas indiqué les actions faites localement ; par exemple avent de répondre OK ou KO, le serveur vérifie qu'un nom de connexion et un mot de passe donnés par le clients sont valides.



Modélisation d'un client Voici un client, élémentaire aussi, sans traitement d'erreurs, synchronisé avec le serveur sur certaines actions ; mais il effectue certaines actions non synchronisées, en local, (`traiterRGet, traiterRput`) par exemple ; on imagine bien la même chose pour `ls`, qui ne figure pas sur l'automate.



Adaptations Vous pouvez aisément adapter cet exercice de modélisation à vos applications imaginées dans le cas de la SAE, en distinguant et listant bien les micro-services de votre serveur, le/les résultats fournis par chacun. Ensuite, pour les clients, selon qu'ils se connectent/déconnectent ou pas, vous aurez ou pas les phases de connexion/déconnexion. Puis, vous modélisez les interactions (appels de micro-services + traitements des résultats prévus).

Quelles sont les hypothèses que vous faites (que vous pouvez faire) par rapport aux déroulements simultanés des clients, dans le cadre le votre projet ?

Informations pratiques de rédaction Utiliser et respecter les règles et consignes de rédaction habituelles. Faire un plan cohérent, avec au minimum une section Introduction, une/des section/s pour le travail présenté, une Conclusion, des références biblio(url)graphiques (outils d'IAs utilisés le cas échéant...chapGPT).

Dans l'introduction présenter bien le contexte de votre travail et ce que vous faites ou devez faire.

Dans la conclusion, faites le bilan de ce qui a été présenté, puis des limitations/ouvertures éventuelles. Entre les deux, valoriser bien le travail qui est fait.

Lorsque vous utilisez des figures/images/tables, ils doivent avoir un titre, les sources citées (quand vous empruntez des images). Les références doivent apparaître dans le texte (sous la forme : dans la Fig. XX..., voir Tab. YY, ...).
