Hypothèses de modélisation et périmètre de l'étude

La modélisation formelle d'un protocole industriel nécessite de définir explicitement un ensemble d'hypothèses et un périmètre d'étude précis. Cette démarche est essentielle afin de maîtriser la complexité du modèle, d'en assurer la cohérence interne et de permettre une interprétation correcte des résultats issus de la vérification formelle [Clarke et al., 1999 ; Sifakis, 2005].

Dans le cadre de ce travail, le modèle UPPAAL a été conçu pour représenter le comportement du protocole Bitbus tel qu'il est utilisé dans un contexte industriel spécifique à EDF, en lien avec les besoins opérationnels identifiés.

4.2.1 Contexte industriel et périmètre fonctionnel

Dans les systèmes industriels critiques, et plus particulièrement dans le domaine nucléaire, les protocoles de communication jouent un rôle central dans l'assurance des fonctions de contrôle-commande.

Les normes de sûreté fonctionnelle insistent sur la nécessité de maîtriser rigoureusement le comportement des échanges, notamment en présence de contraintes temporelles strictes et de situations dégradées (IEC 61508).

Le protocole Bitbus a été conçu pour des environnements industriels nécessitant un comportement déterministe, une gestion explicite des temporisations et une architecture maître–esclave clairement définie (Decotignie, 1993).

Dans le cadre industriel étudié, l'accès direct à un système Bitbus en fonctionnement sur installation réelle n'est pas envisageable, en raison des contraintes de sûreté, de disponibilité et de qualification des équipements propres aux environnements nucléaires (IEC 60880). La modélisation formelle est donc définie de manière à représenter fidèlement les mécanismes fonctionnels observables du protocole, tout en respectant ces contraintes industrielles.

Le modèle se concentre sur les échanges entre un émulateur du maître Bitbus réel, dont le programme est issu de l'extraction du code assembleur contenu dans l'EPROM d'équipements industriels existants (CMX), et une entité esclave abstraite représentative du protocole, désignée par la suite comme générateur de trames. En l'absence de communications possibles en fonctionnement sur tranche, l'entité expérimentale représentant l'esclave ne vise pas à reproduire l'ensemble de ses comportements possibles protocole, notamment en conditions dégradées, mais uniquement son fonctionnement nominal. Elle est utilisée comme moyen

d’observation et de validation empirique, permettant de confronter le comportement observé du maître aux descriptions théoriques issues de la documentation technique, dans une démarche de consolidation de la compréhension du protocole.

Hypothèses sur l’environnement matériel et logiciel

Hypothèses relatives à l’environnement matériel

Dans le cadre industriel étudié, les mécanismes de bas niveau associés à la couche SDLC, tels que le codage et le décodage des trames, le calcul et la vérification du CRC, la gestion des drapeaux ou encore la synchronisation physique sur le bus, sont supposés être pris en charge par une interface matérielle dédiée. Cette hypothèse est conforme aux pratiques industrielles établies, notamment dans les systèmes critiques, où ces fonctions sont généralement implémentées au moyen de composants matériels matures et qualifiés (Storey, 1996).

La modélisation distingue ainsi les mécanismes SDLC ayant un impact structurant sur le déroulement global des échanges Bitbus de ceux dont l’effet est limité à la gestion interne du flux de trames. En particulier, les mécanismes de supervision tels que les trames RR (Receive Ready), RNR (Receive Not Ready), ainsi que la gestion fine des numéros de séquence N(S) et N(R), bien qu’observables sur le bus, n’introduisent pas, dans le contexte étudié, de nouveaux scénarios fonctionnels déterminants au niveau du protocole Bitbus. Leur comportement est donc supposé conforme aux spécifications et correctement assuré par l’interface matérielle SDLC.

À l’inverse, la synchronisation SDLC constitue une phase déterminante du protocole, dans la mesure où elle conditionne l’autorisation même des échanges applicatifs Bitbus. L’établissement ou la perte de cette synchronisation entraîne des décisions explicites du maître, telles que l’activation ou l’interruption des échanges, ainsi que le déclenchement de procédures de reprise.

À ce titre, la synchronisation SDLC est modélisée explicitement, conformément aux recommandations issues des travaux sur les protocoles temps réel et les systèmes distribués critiques (Kopetz, 2011).

Ce choix est cohérent avec les recommandations des normes de sûreté, qui privilégient la délégation des fonctions critiques de bas niveau à des composants matériels certifiés (IEC 61508), tout en concentrant l’analyse formelle sur les comportements pertinents au niveau système.

L'abstraction retenue représente ainsi les interactions SDLC à travers leurs effets fonctionnels sur les échanges Bitbus, sans reproduire l'ensemble des mécanismes internes de la couche liaison.

Hypothèses relatives à l'environnement logiciel

L'environnement logiciel associé au modèle ne vise pas à reproduire exhaustivement l'ensemble de la pile logicielle industrielle réelle, mais à capturer les règles de décision et les mécanismes de réaction du protocole Bitbus face aux événements observables. La modélisation se concentre ainsi sur les comportements logiques et temporels ayant un impact direct sur le déroulement des échanges, tels que la réception ou l'absence de trames, les dépassements de temporisation, ou la détection d'erreurs liées aux intégrités de trames.

Le modèle repose sur une architecture strictement maître–esclave, conforme à l'usage observé du protocole Bitbus dans le système industriel étudié (Decotignie, 1993) ainsi qu'aux éléments techniques fournis par EDF. Les configurations multi-maîtres ou multi-esclaves, bien que prévues par le protocole dans d'autres contextes, ne sont pas considérées dans le cadre de ce travail, afin de maintenir un périmètre de modélisation cohérent avec l'application industrielle ciblée.

Les temporisations intégrées au modèle sont supposées bornées et déterministes. Elles sont issues soit d'observations expérimentales réalisées à l'aide d'un générateur de trames, soit des spécifications techniques mises en place par EDF.

Cette hypothèse est cohérente avec le recours à des automates temporisés et avec les mécanismes de vérification formelle proposés par l'outil UPPAAL (Alur & Dill, 1994 ; Behrmann et al., 2006).

Le comportement de l'entité esclave est modélisé de manière abstraite, sans supposer d'implémentation interne spécifique, mais uniquement à travers les réactions attendues aux sollicitations du maître. Cette modélisation s'appuie sur les mesures réalisées lors des échanges entre l'équipement maître (CMX) et le générateur de trames, ainsi que sur les spécifications relatives au comportement de l'esclave issues de la documentation d'EDF.

Cette approche permet de représenter les interactions essentielles du protocole tout en maintenant un niveau de complexité compatible avec les objectifs de vérification formelle.

Dans cette optique, l'ensemble des choix de modélisation logicielle vise à préserver les décisions fonctionnelles critiques du protocole Bitbus, tout en évitant l'introduction d'une complexité inutile susceptible de conduire à une explosion de l'espace d'états.

Cette réduction contrôlée constitue un levier essentiel pour garantir la faisabilité d'une vérification, conformément aux principes du model-based design appliqués aux systèmes critiques (Holzmann, 1991 ; Sifakis, 2005).

Hypothèses temporelles

Les contraintes temporelles jouent un rôle central dans le fonctionnement du protocole Bitbus, comme c'est le cas pour la majorité des protocoles industriels temps réel.

Dans ce type de systèmes, le respect des délais de communication conditionne directement la cohérence des échanges, la synchronisation des équipements et la capacité du système à détecter et gérer les situations d'erreur.

Les mécanismes temporels constituent ainsi un élément clé pour garantir un comportement déterministe et prévisible du système [Kopetz, 2011].

Dans les architectures maître–esclave, couramment utilisées en automatisme industriel, le temps de polling impose une cadence stricte d'interrogation des équipements, permettant au maître de conserver le contrôle du déroulement des échanges et d'assurer une synchronisation globale du système.

De manière complémentaire, les temporisations de dépassement (*timeouts*) permettent de détecter l'absence de réponse d'un équipement distant et de déclencher des mécanismes de reprise ou de gestion d'erreur.

Ces principes sont largement répandus dans de nombreux protocoles industriels temps réel, tels que Bitbus, Profibus ou Modbus [IEC 61158 ; Tanenbaum & Wetherall, 2011].

Dans ce contexte, le modèle UPPAAL intègre explicitement les principales temporisations observées sur le bus Bitbus, notamment :

- le temps de polling (*t_polling : typiquement = 20ms/2ut*), qui régit la fréquence des interrogations du maître.
- les temporisations de dépassement (*t_out*) *: typiquement = 100ms/10ut*, qui déclenchent des comportements d'erreur ou de reprise en l'absence de réponse.
- les délais de réponse (*t_rep) :réglable mais typiquement* $t_{rep} \in [0, Tout[$, imposant un temps minimal avant l'émission d'une trame de réponse par l'esclave avec :

$$t_{rep} = T_{\{rep\}}^{\{eff\}} = T_{\{rep\}}^{\{prot\}} + \varepsilon$$

Où T_{rep}^{prot} correspond au délai protocolaire volontaire, et ε représente le temps de traitement logiciel nécessaire à l'analyse et à la validation de la trame reçue. On suppose :

$$\varepsilon \ll T_{rep}^{prot}$$

Le terme ε est introduit afin de représenter le coût de traitement logiciel associé à l'analyse, à la validation et au traitement des trames reçues. Il s'agit d'un temps strictement positif, supposé borné et suffisamment faible par rapport à la période de polling, de sorte qu'il n'affecte pas les contraintes temporelles globales du protocole.

Cette hypothèse sera explicitée et justifiée au chapitre 5, consacré à l'implémentation en C, où seront détaillées l'organisation des fonctions d'analyse et de traitement ainsi que les choix d'architecture retenus pour garantir que ce coût reste maîtrisé, déterministe et compatible avec les propriétés temporelles formellement vérifiées.

Cette modélisation renforce le réalisme industriel du modèle et facilite la correspondance entre les contraintes temporelles vérifiées formellement et leur implémentation logicielle effective dans le simulateur, contribuant ainsi à réduire l'écart entre modèle théorique et comportement opérationnel.

Ces paramètres sont considérés comme bornés et déterministes dans le modèle, conformément aux hypothèses classiquement retenues pour la modélisation des systèmes temps réel à l'aide d'automates temporisés [Alur & Dill, 1994 ; Behrmann et al., 2006].

Les valeurs utilisées sont issues de mesures expérimentales et de la configuration du système réel, ce qui permet d'ancrer le modèle dans un comportement temporel effectivement observé et représentatif du fonctionnement industriel.

Limites de la modélisation

Bien que le modèle couvre un ensemble représentatif de scénarios nominaux et dégradés, il ne prétend pas capturer l'intégralité des comportements possibles du système industriel réel.

Certaines situations exceptionnelles, fortement dépendantes de conditions matérielles spécifiques ou de configurations particulières du système, ne sont volontairement pas modélisées.

De même, le modèle ne vise pas à représenter toutes les variations d'implémentation possibles du protocole Bitbus, mais à fournir une représentation cohérente, maîtrisée et vérifiable de son comportement dans le cadre d'usage étudié, en réponse à un besoin industriel identifié par EDF.

Les choix d'abstraction effectués permettent de se concentrer sur les mécanismes fonctionnels et temporels essentiels, sans introduire une complexité excessive qui nuirait à l'analyse formelle.

Cette approche, fondée sur une modélisation partielle mais contrôlée, est largement admise dans les démarches industrielles de vérification formelle, où l'objectif est de raisonner sur les

comportements critiques et les scénarios significatifs plutôt que de viser une exhaustivité irréaliste [Clarke et al., 1999 ; IEC 61508].

Les limites identifiées ne constituent donc pas des lacunes du modèle, mais le résultat d'un choix méthodologique assumé visant à maximiser la valeur démonstrative, la faisabilité de la vérification et l'exploitabilité opérationnelle du modèle, notamment en vue de sa traduction vers un simulateur industriel.

Rôle des hypothèses dans la validité du modèle

L'ensemble des hypothèses et du périmètre définis dans cette section constitue le socle sur lequel repose la validité du modèle UPPAAL.

En explicitant clairement les choix de modélisation, il devient possible de relier les comportements observés dans le modèle aux situations réelles rencontrées sur le terrain, et d'interpréter correctement les résultats issus de la vérification formelle.

Cette abstraction est essentielle pour établir une continuité logique entre le modèle formel, les propriétés temporelles et fonctionnelles vérifiées, et le simulateur développé dans la suite du travail.

Elle permet notamment de démontrer que les comportements reproduits par le simulateur reposent sur un modèle formel dont les hypothèses sont explicitement identifiées et rigoureusement maîtrisées, conformément aux principes du *model-based design* (Sifakis, 2005). L'ensemble des hypothèses retenues pour la modélisation du protocole Bitbus est synthétisé dans le **Tableau 1**, offrant une vue d'ensemble claire et structurée de leur rôle et de leur impact sur le modèle.

Synthèse :

En synthèse, les hypothèses définies dans cette section établissent un cadre de modélisation maîtrisé permettant de représenter fidèlement les mécanismes critiques du protocole Bitbus tout en garantissant la faisabilité de la vérification formelle.
Elles constituent un compromis méthodologique assumé entre réalisme industriel et maîtrise de la complexité, en isolant les comportements essentiels à l'analyse de sûreté et de robustesse.
Ce cadre définit ainsi un contrat explicite entre le système réel, le modèle UPPAAL et le simulateur dérivé, assurant la cohérence globale de la démarche de modélisation adoptée dans ce travail.

Tableau 1 : L'ensemble des hypothèses retenues pour la modélisation du protocole Bitbus :

Catégorie	Hypothèse de modélisation	Justification industrielle et méthodologique	Impact sur le modèle formel (UPPAAL)
Architecture	Architecture maître–esclave stricte	Usage effectif du protocole Bitbus dans le système étudié (EDF)	Simplification de la structure globale
Environnement matériel	Mécanismes SDLC gérés matériellement	Fonctions assurées par composants certifiés	Focus sur les effets fonctionnels
Synchronisation SDLC	Synchronisation SDLC modélisée explicitement	Conditionne l'autorisation des échanges Bitbus	Ajout d'états et transitions de synchronisation
Supervision SDLC	RR, RNR et N(S)/N(R) abstraits	Pas de nouveaux scénarios majeurs	Réduction de l'espace d'états
Temporisations	Temporisations bornées et déterministes	Observations expérimentales et spécifications EDF	Horloges temporisées UPPAAL
Maître Bitbus	Programme maître issu de l'EPROM (CMX)	Code réel extrait et testé hors tranche	Modélisation fidèle du maître
Esclave Bitbus	Esclave modélisé par un générateur de trames	Émulation des réponses protocolaires	Automate abstrait réactif
Cadre expérimental	Tests hors tranche sur banc dédié	Contexte sécurisé et reproductible	Scénarios contrôlés en laboratoire
Environnement logiciel	Simplification de la pile logicielle	Concentration sur les règles du protocole	Abstraction des couches non critiques
Événements observables	Événements observables modélisés	Réceptions, timeouts, erreurs	Transitions fonctionnelles clés
Objectif global	Compromis fidélité / faisabilité	Maîtrise de l'explosion d'états	Vérification possible