RED HAT®
CONSULTING

redhat.

LIVRE BLANC

UNE APPROCHE DE CONCEPTION STRUCTURÉE POUR L'AUTOMATISATION DES PROCESSUS MÉTIER

Les nouvelles infrastructures et approches de développement logiciel apportent des outils qui permettent de moderniser les systèmes d'automatisation des processus métier. Avec une approche de conception qui se fonde sur l'application de bonnes pratiques, du développement des logiciels modernes à l'automatisation des processus métier, les entreprises peuvent adopter des pratiques d'ingénierie rigoureuses tout en limitant les risques.

facebook.com/redhatinc
@RedHat_France
linkedin.com/company/red-hat

fr.redhat.com

SYNTHÈSE

Dans de nombreuses grandes entreprises, le service informatique a pour mission de fournir à l'équipe métier des outils qui lui permettent de faire face à la concurrence, de respecter les réglementations du secteur et de fidéliser la clientèle. Le service informatique est censé proposer des solutions de qualité, suffisamment flexibles pour répondre à des changements rapides et fréquents et dont les coûts sont à la fois prévisibles et, de préférence, peu élevés.

Il existe deux méthodes qui permettent à toute une équipe métier d'obtenir la solution idéale, au bon moment : le développement de logiciels personnalisés et l'automatisation des processus métier. Ces deux approches présentent chacune leurs avantages et leurs inconvénients. D'un côté, le développement logiciel est une discipline d'ingénierie aux processus rigoureux, mais généralement lents. De l'autre, l'automatisation des processus métier réduit le délai de mise sur le marché en donnant aux non-initiés la possibilité de codifier la logique métier, avec, en contrepartie, une augmentation des risques d'erreur.

Les nouvelles infrastructures et approches de développement logiciel apportent des outils qui permettent de moderniser les systèmes d'automatisation des processus métier. Avec une approche de conception qui se fonde sur l'application de bonnes pratiques, du développement des logiciels modernes à l'automatisation des processus métier, les entreprises peuvent adopter des pratiques d'ingénierie rigoureuses tout en limitant les risques. En associant le développement logiciel moderne à cette approche de l'automatisation des processus métier, les entreprises peuvent confier la codification de la logique métier à des experts afin d'obtenir de meilleurs résultats, des délais de mise sur le marché plus courts et des coûts prévisibles plus bas.

L'ÉVOLUTION DE L'AUTOMATISATION DES PROCESSUS MÉTIER ET DU DÉVELOPPEMENT LOGICIEL

Les responsables informatiques recherchent sans cesse de nouveaux produits ou de nouvelles technologies susceptibles de les aider à répondre à la demande de l'équipe métier. Malheureusement, la plupart des logiciels commerciaux ne permettent pas aux entreprises d'innover assez rapidement ou de se démarquer suffisamment face à la concurrence. Résultat : l'équipe informatique se voit confier le développement de logiciels personnalisés.

Auparavant, le développement d'un logiciel était un processus lent et complexe qui nécessitait une communication efficace, dès le début, entre les experts métier et les développeurs. Souvent, l'entreprise adoptait une méthode en cascade : l'expert métier donnait ses directives au développeur en amont, qui concevait, développait, puis testait ensuite l'application. Or, avec cette méthode, l'expert métier ne pouvait évaluer l'application qu'une fois celle-ci terminée. Chaque changement était donc compliqué et coûteux.

L'automatisation des processus métier (ou parfois simplement automatisation métier) constitue une solution alternative au développement logiciel dirigé par le service informatique. En effet, cette méthode permet aux experts métier de définir et d'exécuter, de manière rapide et itérative, la logique métier dont ils ont besoin. Dans ce cas, la logique métier se traduit par des ensembles de processus, de règles ou de workflows, plutôt que par des fonctions applicatives. Ces règles sont ensuite exécutées par un moteur en réponse à certaines conditions ou à certains événements.

Prenons un exemple : le processus d'intégration des nouveaux clients dans un établissement financier est régi par chaque gouvernement à tous les niveaux, afin de lutter contre le blanchiment d'argent et toute autre activité illégale. Cependant, les réglementations en la matière ne cessent d'évoluer. Avec l'automatisation des processus métier, le spécialiste de la conformité d'un établissement financier peut définir des conditions et des règles qui appliquent la nouvelle réglementation. Il peut ensuite déployer ces règles dans un moteur sans l'aide d'un architecte, d'un développeur ou d'un autre membre du service informatique.

Bien que l'automatisation des processus métier et le développement de logiciels personnalisés partagent le même objectif, à savoir encoder la logique métier dans un système logiciel, les deux approches ont été développées séparément. L'environnement informatique isolé et fragmenté alors créé engendrait des coûts plus élevés à cause de la duplication des serveurs, des systèmes d'exploitation et d'autres licences logicielles, de la maintenance, de l'assistance, etc. Et en évitant les contraintes du développement logiciel, notamment les pratiques d'ingénierie rigoureuses, il se peut que l'automatisation des processus métier ait augmenté les risques d'erreur. Par exemple, certains principes d'ingénierie logicielle, comme la tenue d'un historique des modifications pour le contrôle des versions, le suivi des tests et des versions prêtes à entrer en production, ainsi que la gestion des configurations (notamment les cycles de vie de nombreux composants) n'étaient pas forcément respectés dans les environnements d'automatisation métier.

Au cours des dernières décennies, la conception et le développement de logiciels ont considérablement évolué. Dans de nombreux cas, cette évolution se basait sur les mêmes critiques à l'origine de l'essor de l'automatisation des processus métier. L'architecture logicielle a depuis progressé, passant d'une conception basée sur d'énormes applications monolithiques (longues à construire et difficiles à modifier) à une architecture basée sur des services, avec des composants applicatifs conçus dans un but unique, plus faciles à créer, à mettre à jour et à maintenir.

Les méthodes de développement logiciel ont elles aussi évolué. La méthode en cascade classique se caractérise par une planification minutieuse et des phases bien distinctes, qui complexifient la reproduction de certaines parties des processus. Cette méthode a été remplacée par des approches itératives qui visent à produire rapidement des applications fonctionnelles, puis à les réviser avec l'expert et à effectuer les changements à chacune des nombreuses itérations. Ces structures, telles que l'« extreme programming », Scrum et Scaled Agile Framework, se sont améliorées avec le temps, en parallèle des avancées technologiques. Aujourd'hui, les pratiques d'intégration et de distribution continues (CI/CD) proposent une approche rapide et itérative du déploiement. Ces pratiques s'appuient sur l'idée que le code doit être intégré plusieurs fois par jour et testé tout au long du développement afin de respecter le délai de lancement du logiciel.

UNE APPROCHE DE CONCEPTION MODERNE DE L'AUTOMATISATION DES PROCESSUS MÉTIER

De nombreuses solutions d'automatisation des processus métier n'ont pas su tirer parti des progrès en matière de conception et de développement de logiciels. Une approche efficace, qui fait bénéficier l'automatisation des processus métier des avantages du développement logiciel moderne, organise les différents aspects de la conception en ensembles de composants architecturaux qui nécessitent au moins un choix de mise en œuvre clé. Dans un environnement construit selon cette approche, une entreprise peut utiliser la même infrastructure pour le développement logiciel et pour l'automatisation des processus métier.

Les composants architecturaux constituent les activités principales d'une stratégie efficace d'automatisation des processus métier :

1. **Modélisation** de la logique métier et test des scénarios qui la définissent en développant un code applicatif ou en définissant les éléments de base de l'automatisation des processus métier : règles, événements, processus et workflows
2. **Gestion des versions** des artefacts du modèle, avec notamment un historique vérifiable des modifications
3. **Construction** d'un paquet déployable à partir d'un ensemble d'artefacts, test d'automatisation et historique du paquet
4. **Stockage** du paquet dans un référentiel principal
5. **Déploiement** et test du paquet dans un environnement d'exécution, tel qu'un serveur d'applications pour le code ou un moteur pour les éléments d'automatisation métier
6. **Exécution** de la logique métier

Les composants architecturaux et les principaux choix de mise en œuvre sont décrits ci-dessous. La complexité, les capacités et le coût d'un environnement dépendent des décisions prises au niveau de chaque composant.

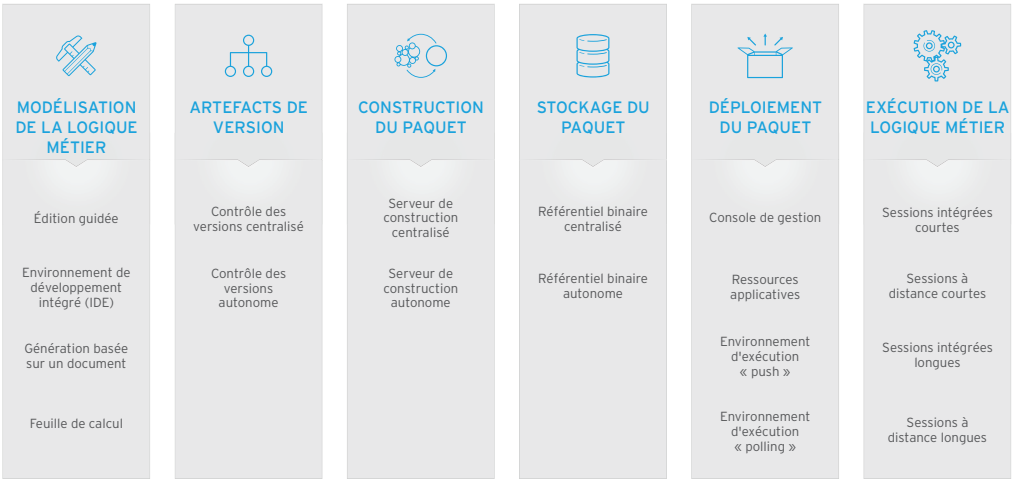


Figure 1 : composants architecturaux et choix de mise en œuvre correspondants pour l'automatisation des processus métier

1. MODÉLISATION

La modélisation consiste à capturer la logique métier en vue de son exécution. En général, les utilisateurs modélisent la logique métier sur la base de leur connaissance du comportement attendu du système. La logique métier peut être modélisée à l'aide d'un code applicatif, d'éléments d'automatisation métier ou d'un mélange des deux.

Les programmeurs qui optent pour le code applicatif modélisent la logique métier en utilisant un langage de programmation. La plupart des programmeurs rédigent leur code dans un environnement de développement intégré (IDE), mais tout type d'éditeur de texte peut être utilisé, du moment qu'il produit un code conforme à la norme.

Avec l'automatisation des processus métier, ce sont les experts métier qui modélisent la logique métier en créant des règles, des processus, des événements et des workflows à l'aide de diverses méthodes :

- **Outil d'édition guidée :** un outil d'édition guidée offre une interface intuitive qui masque les détails techniques. Son utilisation ne nécessite donc aucune connaissance technique. Cet outil permet à l'utilisateur de définir des scénarios de test pour valider l'exactitude des éléments. L'édition guidée remplit ainsi l'objectif original de l'automatisation des processus métier en fournissant un outil qui permet aux experts métier de réaliser des modélisations sans l'aide d'un programmeur.
- **Feuilles de calcul :** les experts métier peuvent aussi utiliser des feuilles de calcul pour définir des règles sous la forme de tables de décisions, notamment lorsqu'il y a des répétitions entre les lignes. La plupart du temps, les experts métier savent utiliser les feuilles de calcul et peuvent donc facilement modifier la valeur de chaque ligne pour définir la logique voulue. Les feuilles de calcul peuvent aussi servir à modéliser des processus.
- **Génération basée sur un document :** selon cette approche, un modèle définit la structure d'une règle ou d'un processus. Les données métier contenues dans un document (un contrat ou une offre promotionnelle, par exemple) sont appliquées au modèle en vue de générer des règles ou des processus.
- **Environnements de développement intégrés (IDE) :** les programmeurs utilisent des environnements de développement intégrés (IDE) pour rédiger le code applicatif. Ces environnements peuvent également servir pour créer des éléments d'automatisation métier. Pour créer des règles métier et des diagrammes de processus, l'utilisateur doit ajouter un plug-in à son IDE. Les IDE intègrent d'autres fonctions, telles que la coloration syntaxique, et offrent un environnement de travail plus familier aux utilisateurs plus techniques.

Une fois la logique métier modélisée, les experts métier peuvent utiliser les pratiques de développement dirigé par le comportement pour guider le développement et les futurs tests de l'application. Avec ce type de pratiques de développement, les experts métier peuvent créer de courts scénarios qui décrivent le comportement souhaité de l'application dans un langage compréhensible et qui définissent ses exigences. Ces scénarios sont ensuite utilisés par les développeurs (ou les experts métier dans une approche d'automatisation des processus métier) pour modéliser la logique métier. À des stades plus avancés du cycle de développement, ces mêmes scénarios servent de base pour les tests automatisés de l'application.

2. GESTION DES VERSIONS

Le système de contrôle des versions est le référentiel principal des artefacts de logique métier. Que la logique métier soit représentée sous la forme d'un code applicatif ou de règles métier, et quel que soit l'outil utilisé, les artefacts créés doivent ensuite être enregistrés. La gestion des versions ne se limite pas à l'enregistrement de l'artefact, elle vise aussi à documenter toutes les modifications apportées ainsi que leur raison et à permettre la récupération d'une version antérieure spécifique. La gestion des versions permet aussi de suivre plusieurs séquences différentes de modifications apportées à un même artefact.

En général, les produits d'automatisation métier utilisent des bases de données pour enregistrer les versions des artefacts issus de l'automatisation des processus métier. Cependant, ces bases de données ne présentent malheureusement pas tous les avantages d'un véritable outil de gestion des versions. Les systèmes de contrôle des versions standard basés sur les fichiers, tels que Git, Mercurial ou Apache subversion, sont fréquemment utilisés pour le code applicatif. De nombreuses entreprises s'en servent déjà pour créer des systèmes de contrôle des versions partagés et centralisés. Lorsque le système de contrôle des versions existant est utilisé, les

programmeurs et les experts métier n'ont pas besoin de se préoccuper des questions telles que la sécurité, la conservation des données et la sauvegarde. Toutefois, si une entreprise ne dispose pas d'une infrastructure de gestion des versions centralisée ou si sa mise en œuvre se révèle trop complexe ou trop coûteuse, elle peut toujours enregistrer ses artefacts de logique métier dans un système de contrôle des versions autonome et dédié.

3. CONSTRUCTION

Le système de construction est le processus principal pour la préparation et l'assemblage des artefacts dans un paquet en vue de leur futur lancement. Il est important que la mise en paquet des artefacts soit efficace et cohérente, idéalement sans intervention humaine ou avec une intervention minimale.

Les paquets sont souvent créés à l'aide d'un outil d'automatisation de la construction, tel qu'Apache Maven. L'outil d'automatisation de la construction est configuré non seulement pour exécuter le nombre de commandes nécessaires à l'obtention des artefacts à partir du système de contrôle des versions, mais aussi pour préparer et assembler les artefacts en vue de leur futur lancement. Il est possible d'automatiser la construction afin de créer un nouveau paquet chaque fois qu'un artefact est enregistré dans le système de contrôle des versions.

C'est à cette étape qu'a lieu le test automatisé du nouveau paquet, dans le cadre de l'intégration continue. Cette phase de test permet d'identifier immédiatement toute erreur d'intégration qui peut alors être corrigée avant la création de la version comprenant la suite du code et la création de nouveaux paquets. Ainsi, un paquet qui contient des erreurs d'intégration ne doit pas dépasser cette étape du processus de lancement.

Comme pour le système de contrôle des versions, de nombreuses entreprises disposent déjà d'un système d'automatisation de la construction, et lorsque ce système est exploité, les utilisateurs n'ont pas à se préoccuper de son installation, de sa configuration ni de son maintien. De plus, les ressources logicielles en paquet doivent souvent respecter les politiques des entreprises et les réglementations externes, qui sont déjà incluses dans le système de construction centralisé.

Si le système de construction centralisé ne prend pas en charge Maven ou s'il existe d'autres problèmes qui empêchent l'utilisation du système existant, il est toujours possible de recourir à un système de construction distinct. Par contre, la création d'un système de construction distinct multiplie le nombre des systèmes de construction, par exemple, un par équipe d'application, voire un par programmeur et expert métier.

4. STOCKAGE

Le référentiel de paquets est la source principale des paquets prêts à être déployés. Les paquets logiciels sont souvent déployés plusieurs fois sur différents serveurs, dans différents datacenters. Il est plus efficace et cohérent de déployer des paquets stockés dans un référentiel que de reconstruire le paquet à chaque déploiement. Cela permet aussi de s'assurer que le paquet déployé est toujours le même.

Comme nous l'avons déjà vu, les entreprises utilisent souvent un référentiel centralisé pour satisfaire les exigences en matière de sécurité, de gouvernance, de conformité réglementaire, de conservation des données et d'audit. Encore une fois, il est aussi possible d'utiliser un référentiel autonome compatible avec Maven, même si cela implique la multiplication des référentiels au sein de l'entreprise.

5. DÉPLOIEMENT

Les paquets logiciels sont déployés depuis le référentiel, dans un environnement d'exécution qui dépend du type de paquet. Par exemple, une application exécutable binaire et autonome sera copiée sur le serveur où elle sera exécutée, tandis qu'une application Java d'entreprise peut être chargée sur un serveur d'applications. Les paquets d'automatisation métier sont déployés dans le moteur d'automatisation des processus métier, c'est-à-dire l'environnement d'exécution des règles et des processus.

Les paquets peuvent être déployés sans intervention humaine, ou avec une intervention minimale. Même les étapes de déploiement initiées par un utilisateur sont prédéfinies et validées en amont afin de réduire le risque d'erreur. Les règles de déploiement, telles que les priorités et le calendrier de déploiement, peuvent aussi être formalisées.

Elles sont également utilisées pour définir les tests à effectuer avant le lancement. Avant son déploiement en production, un paquet est habituellement déployé dans des environnements d'assurance qualité et de préproduction en vue d'être testé. Les tests réalisés sont plus complets que les tests automatisés réalisés à l'étape de la construction. Par exemple, lors de la phase d'assurance qualité, le testeur va agir comme un utilisateur qui découvre de nouvelles fonctions, alors qu'en préproduction, le testeur effectue surtout des tests de performances et de charge. Les paquets ne seront déployés en production que s'ils passent avec succès les tests réalisés dans ces environnements.

Le plus souvent, les paquets sont déployés selon la méthode « push » qui consiste à déployer le paquet dans l'environnement d'exécution via un processus automatisé. Avec l'approche de distribution continue, le déploiement s'effectue une fois que le paquet est mis à jour dans le référentiel, mais il est possible de le configurer pour qu'il respecte un calendrier. Chaque environnement d'exécution reçoit la dernière version du paquet conformément aux règles de déploiement. Ainsi, l'utilisateur n'a plus besoin de s'en préoccuper.

Une autre approche similaire consiste à utiliser une console de gestion (en général une application web avec une interface utilisateur) pour déployer les paquets. L'utilisateur configure les paramètres de déploiement du paquet, tels que le nom, la version et l'environnement d'exécution cible, puis lance le déploiement manuellement. Ensuite, la console de gestion « pousse » le paquet vers l'environnement d'exécution. Ici, ce processus n'est pas automatisé et requiert l'intervention d'un utilisateur.

Il est aussi possible de recourir à une approche dite « pull » ou « polling ». Avec cette approche, l'environnement d'exécution analyse et récupère les paquets mis à jour dans le référentiel. Malheureusement, elle augmente la complexité en renforçant le lien entre les étapes de construction et de déploiement, car le paquet mis à jour est récupéré juste après sa construction. Par exemple, si plusieurs environnements d'exécution (développement, test, production) utilisent le même référentiel, ils vont tous récupérer la dernière version. Même s'il existe plusieurs manières d'atteindre cet objectif, il est souvent préférable d'utiliser les mécanismes « push ».

Pour le déploiement des artefacts issus de l'automatisation des processus métier, il est aussi possible d'opter pour une approche basée sur les ressources applicatives. Avec cette approche, les artefacts sont mis en paquet avec le moteur d'exécution dans l'application. Il s'agit d'une approche simple et unifiée, mais elle génère une application monolithique. Le déploiement ressemble alors plus à celui d'une application classique qu'à celui d'un paquet d'automatisation métier. Dans l'idéal, cette approche est associée à une autre approche impliquant une console de gestion ou la méthode « push ».

6. EXÉCUTION

En règle générale, l'environnement d'exécution comprend tous les serveurs, systèmes de stockage, systèmes d'exploitation, réseaux et autres systèmes nécessaires à l'exécution de la logique métier. Dans ce document, nous aborderons uniquement le composant d'automatisation métier.

Les aspects clés de mise en œuvre de l'environnement d'exécution sont différents de ceux des autres composants architecturaux de l'approche décrite ici. L'étape de modélisation dépend des compétences de l'utilisateur ainsi que de ses préférences personnelles, tandis que pour les étapes de gestion des versions, de construction et de stockage, il n'a qu'à choisir l'emplacement du référentiel, qui peut être modifié à tout moment. Les choix de l'utilisateur pendant l'étape de déploiement affectent la manière dont les paquets sont placés dans l'environnement d'exécution, mais pas la manière dont ils sont exécutés.

Les aspects techniques de l'environnement d'exécution ont une influence sur l'exécution de la logique métier, la gestion des performances, l'évolutivité et la fiabilité. Les décisions liées à la conception de cet environnement peuvent être déterminantes, tant au niveau de la satisfaction client que de l'efficacité de l'environnement.

L'une des décisions les plus importantes concerne le moteur d'automatisation des processus métier, qui peut être soit distribué comme un élément de l'application, soit établi en tant que service partagé et centralisé. Avec l'exécution intégrée, le moteur fait partie de l'application et son utilisation est dédiée à cette application. L'application et son moteur sont exécutés sur la même machine virtuelle Java™ afin de fournir des performances optimisées. Malheureusement, ce type d'exécution coûte plus cher, car davantage de ressources de calcul sont alors nécessaires.

Avec l'établissement d'un service partagé et centralisé, la logique applicative est séparée du moteur d'automatisation des processus métier, ainsi que des règles et des processus qui y sont déployés. Un moteur unique, et donc cohérent, est disponible pour toutes les applications au sein de l'entreprise. Ainsi, les applications requièrent moins de ressources de calcul et il est possible de faire évoluer le moteur partagé selon les besoins pour prendre en charge de nombreuses applications. Cette approche nécessite des techniques de programmation distribuées, mais elle permet d'accéder au moteur via plusieurs protocoles. Celui-ci n'est donc pas réservé aux applications Java.

Une autre décision clé concerne l'interaction entre les applications et le moteur. Certaines applications exécutent une règle et renvoient un résultat dans le cadre d'une interaction unique. Pour ces sessions courtes, le moteur ne stocke aucune donnée applicative après l'exécution de la règle. En général, l'application gère les données dans un système de stockage persistant distinct et les transfère au moteur au moment opportun.

D'autres applications exécutent des processus métier qui peuvent s'étendre sur plusieurs jours, semaines, voire plus. Lors de ces interactions, l'application crée une session longue avec le moteur et celui-ci gère la persistance des données applicatives à utiliser dans les étapes suivantes. Dans ce cas, les données sont également accessibles par d'autres applications.

La durée de la session et la gestion de l'état peuvent significativement affecter les performances et l'évolutivité de l'environnement d'exécution. Les sessions courtes peuvent évoluer horizontalement sans complexifier l'environnement, alors que les sessions longues peuvent évoluer à l'horizontale comme à la verticale, mais pas sans une base de données bien conçue. En général, les sessions longues ne sont utilisées que si elles sont vraiment nécessaires, comme pour la gestion des processus métier, car elles requièrent beaucoup de ressources.

CONCLUSION

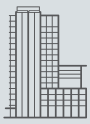
L'automatisation des processus métier a été pensée pour pallier l'incapacité du développement logiciel à répondre aux besoins métier actuels. Toutefois, pour y parvenir, l'automatisation des processus métier augmente les risques et introduit d'autres difficultés. Au cours de ces 20 dernières années, le rythme des innovations s'est emballé et les solutions d'automatisation des processus métier n'ont pas su y faire face. L'économie est aujourd'hui marquée par le bouleversement numérique des marchés matures ainsi que par une complexification de l'environnement réglementaire.

Les recherches menées dans le secteur, telles que le rapport « 2015 State of DevOps Report » de Puppet[1], démontrent que les éditeurs de logiciels qui affichent les meilleures performances sont ceux qui sont les mieux préparés à affronter ce marché volatil. C'est pour cette raison qu'il est important d'intégrer des solutions d'automatisation des processus métier dans les infrastructures et approches modernes du développement et du déploiement de logiciels.

Les services de consulting Red Hat® proposent des pratiques d'automatisation des processus métier qui se fondent sur l'approche unifiée présentée dans ce livre blanc. L'objectif est d'aider les entreprises à tirer parti de l'automatisation métier en exploitant les techniques de développement logiciel moderne plutôt qu'en les évitant.

Pour en savoir plus sur les pratiques d'automatisation des processus métier proposées par les services de consulting Red Hat, vous pouvez consulter la fiche technique disponible à la page redhat.com/fr/resources/red-hat-consulting-discovery-session-business-automation.

1 https://puppet.com/resources/white-paper/2015-state-of-devops-report



À PROPOS DE RED HAT

Premier éditeur mondial de solutions Open Source, Red Hat s'appuie sur une approche communautaire pour fournir des technologies Linux, de cloud, de virtualisation, de stockage et de middleware fiables et performantes. Red Hat propose également des services d'assistance, de formation et de consulting reconnus. Situé au cœur d'un réseau mondial d'entreprises, de partenaires et de communautés Open Source, Red Hat participe à la création de technologies novatrices qui permettent de libérer des ressources pour la croissance et de préparer ses clients au futur de l'informatique.

facebook.com/redhatinc
@RedHat_France
linkedin.com/company/red-hat

EUROPE, MOYEN-ORIENT ET AFRIQUE (EMEA)
00800 7334 2835
fr.redhat.com
europe@redhat.com

TURQUIE
00800-448820640

ISRAËL
1-809 449548

ÉAU
8000-4449549

fr.redhat.com
#INC0410962_v1_201606_KVM