Future Generation Computer Systems 102 (2020) 95–111

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Quantifying the adaptability of workflow-based service compositions

Khavee Agustus Botangen [a,*], Jian Yu [a], Yanbo Han [b], Quan Z. Sheng [c], Jun Han [d]

[a] Auckland University of Technology, Auckland, New Zealand
[b] Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, China
[c] Macquarie University, Sydney, Australia
[d] Swinburne University of Technology, Melbourne, Australia

HIGHLIGHTS

- Adaptability measurement is concerned with the variability in both process workflow and binding to partner services.
- Adaptability is compared regardless of process size and specification framework.
- Process specification frameworks are compared based on the maximum process adaptability they can offer.
- The measurement raises designers' awareness on adaptability, to facilitate design decisions towards composing quality systems.

ARTICLE INFO

Article history:
Received 19 January 2019
Received in revised form 29 July 2019
Accepted 6 August 2019
Available online 8 August 2019

Keywords:
Adaptability
Metrics
Service composition
Web services
WS-BPEL

ABSTRACT

Service composition, which is the integration of heterogeneous Web-accessible services, has introduced an open phenomenon for creating software systems that are deployed in an unpredictable execution context, where changes in the requirements, user preferences, and component services frequently occur. The consideration of adaptability in designing such systems is necessary in this unstable and volatile environment. There have been intensive efforts on solutions to make service compositions adaptable, but little attention is given to the evaluation of such adaptability. This paper proposes a set of metrics to quantify the adaptability of service compositions that are specified from the adaptive WS-BPEL-based frameworks. The metrics are based on two adaptability dimensions: the *structure* and *binding* variabilities. Structure variability, enables runtime changes to the composition workflow while binding variability, allows dynamic binding to concrete partner services. We evaluate the metrics through a case study, utilising variants of a travel booking process specified from different adaptive WS-BPEL-based frameworks found in the literature. The metrics are applicable to a wide spectrum of adaptive service compositions and their frameworks. A support tool is also developed to automate metrics computation. Through the metrics, adaptability of service compositions and their frameworks can be assessed and compared. This would facilitate design decisions in building flexible and robust services where adaptability is a first class concern.

1. Introduction

The Internet has enabled the largest computing platform, *i.e.*, the Web, where a functionality can be published as a Web-accessible service *a.k.a. Web service*. New and value-added applications, called *service compositions* or *composite services* [1], have been developed by integrating existing Web services to fulfil more complex user goals. Web services are rapidly increasing,[1] which is attributed to the recently widespread adoption of service computing, compelled by the expansion of services to both social and economic sectors of the society [2]. Service composition has therefore become a promising development model for modern distributed systems, which accordingly are called *service* (-based) *systems* [2,3].

Adaptability is widely acknowledged as an inherent quality attribute to service-based systems, and has become a major challenge in the design of such systems [3–7]. However, the immense focus on the development and classification of approaches to make systems adaptable has left little attention on how to evaluate adaptability which we believe is also of equal importance. Adaptability allows a system to continuously offer the required functionality and quality of service in a dynamic context, where requirements and the environment constantly change. With adaptability, a composite service can change its process flow

* Corresponding author.
E-mail addresses: khavee.botangen@aut.ac.nz (K.A. Botangen), jian.yu@aut.ac.nz (J. Yu), yhan@ict.ac.cn (Y. Han), michael.sheng@mq.edu.au (Q.Z. Sheng), jhan@swin.edu.au (J. Han).

[1] In the 2nd quarter of 2019, there are already over 21,600 Web services registered in a public directory https://www.programmableweb.com/.

https://doi.org/10.1016/j.future.2019.08.010
0167-739X/

logic, or partner services at runtime for a particular context of use. As opposed to traditional monolithic systems, service compositions consider the crucial role of adaptability for achieving an acceptable system utility, *i.e.*, the measure of a composition's quality [8]. Adaptability, can positively or negatively affect other quality attributes [9]. Hence its design, implementation, and evaluation in the life cycle of service compositions has become a significant undertaking. Towards such goal, an important step is to *quantify* adaptability. This can raise developers' and designers' awareness, and aid their decisions towards composing adaptable systems. A concrete measurement will enable adaptability to be considered against other quality attributes during design trade-offs. Besides, unlike other quality attributes such as performance and reliability, there has been little work on adaptability metrics, thus, little attention has been given to adaptability, in systems design trade-offs [7].

In this paper, we propose metrics to quantify the adaptability of service compositions, particularly those specified from the adaptive BPEL-based frameworks. The metrics specifically accommodate two prominent adaptability dimensions: (*i*) *structure variability* — the variability of a process[2] to adapt to runtime changes in the composition logic, resulting to changes in the process structure; and (*ii*) *binding variability* — the variability for a process to dynamically select the most suitable concrete service. Quantifying adaptability could yield several benefits to service compositions. On the one hand, it allows designers to evaluate the degree of process adaptability before or during development, e.g., designers can get direct feedback for changes they introduce, making them aware of changes that limit adaptability. Consequently, it can improve the overall design of a process especially if used in conjunction with evaluating other quality attributes. On the other hand, adaptability of process configurations can be compared between processes specified from the same or different frameworks. This enables designers to evaluate alternative process configurations and decide the most suitable to their adaptability requirements and development priorities. Moreover, our proposed metrics extend to quantifying the adaptability of the specification frameworks. This facilitates the designer's decision on the framework to use in specifying a certain process. We illustrate and evaluate our approach through a travel booking process case study where we examine various representative frameworks that respectively implement existing BPEL-based service composition adaptability mechanisms.

This paper extends our earlier publication on the topic [10] and we highlight the following major contributions:

- First, we provide a complete definition of the metrics to quantify and compare adaptability of BPEL processes. The metrics definitions are based on the two underlying adaptability dimensions: structure variability and binding variability.
- Second, we demonstrate the applicability of the metrics to the wide spectrum of BPEL-based service composition frameworks. We use the metrics to evaluate the adaptability of processes specified from various BPEL-based frameworks regardless of the implemented adaptability mechanism.
- Third, we expand the use of the metrics not just with processes but also with frameworks. Specification frameworks are compared based on the maximal adaptability that they can provide for a certain process.
- Lastly, we develop a support tool to automate the calculation of the metrics. Likewise, we evaluate the metrics in terms of discriminative power and correlation to process size.

[2] We also use the term *process* to refer to *service composition*.

In the remainder of the paper, Section 2 presents the underlying concepts and Section 3 defines the metrics. Section 4 examines the application of the metrics in various adaptive service compositions and frameworks. Section 5 presents the comparison of processes and frameworks, inclusion of adaptability in design decisions, and the limitations of this work. Section 6 describes the implementation tool we have developed and the evaluation of the metrics' comparability property. Section 7 presents a literature survey that exemplifies various adaptability metrics and their perspectives of adaptability, and Section 8 concludes the paper.

2. Metrics rationale

In this section, we introduce the concepts underlying our proposed adaptability metrics. We discuss the basic elements of a BPEL specification and the major adaptability mechanisms implemented in BPEL. Then, we present details about the two adaptability dimensions considered in the metrics.

2.1. WS-BPEL and BPEL Processes

With the emerging service technologies, the Web Services Business Process Execution Language (WS-BPEL or BPEL) has become a popular standard[3] to specify enterprise level service compositions [11]. It supports the orchestration not just of the conventional XML-based or SOAP-based Web services but also services using other interfaces or protocols such as RESTful Web services [12] and OGC Web services [13].

Specifying a BPEL composition is essentially defining a new Web service that is a composition of existing services. A BPEL composition shows how multiple service interactions with partner Web services are coordinated to achieve a business goal, and the logic necessary for this coordination [14]. The standard BPEL specification describes the control flow and data flow dependencies among the interacting partners (clients and Web services) forming the composition [15,16]. The control flow and data flow, respectively describe the ordering of interactions and exchange of data. BPEL specification is mainly composed of elements categorised into *atomic* or *structured* activities. The atomic activities, *a.k.a.* basic activities, are the ⟨*invoke*⟩, ⟨*receive*⟩, ⟨*reply*⟩, and ⟨*assign*⟩, which respectively invokes a partner Web service, receives message from a client, generates responses for synchronous operations, and manipulates data variables. The activities ⟨*invoke*⟩, ⟨*receive*⟩, and ⟨*reply*⟩ are called messaging activities for their use in either one-way (asynchronous) or request/response (synchronous) messaging operations. Meanwhile, a structured activity defines the execution flow among the atomic and structured activities within it. Common structured activities are the ⟨*sequence*⟩, ⟨*switch*⟩, and ⟨*while*⟩, each of which defines a basic execution control. Other structured activities are the ⟨*flow*⟩, which defines synchronisation and concurrency, and ⟨*pick*⟩, which selects an execution choice subject to an external event, (*i.e.,* a message or alarm event). A BPEL specification, (*a.k.a.* a BPEL process), is deployed and executed on BPEL compliant orchestration engines. Generally, BPEL describes a workflow-oriented composition model for the behaviour of a business process based on interactions between the process and its partner Web services.

We visualise a BPEL process using a hierarchical tree structure considering the activities as nodes, and the relationships among activities (control flow) as edges, e.g., Fig. 2. In the subsequent sections, we show how this structural representation including the logical behaviour of each activity becomes relevant to our metrics.

[3] BPEL has been supported by the industry and open-source community as shown by the development of some popular BPEL engines: *e.g.*, IBM WebSphere Process Server, Oracle BPEL Process Manager (now part of Oracle SOA Suite), Microsoft BizTalk Server, SAP Exchange Infrastructure, and Apache ODE.

2.2. Extending BPEL for dynamic adaptability

The standard BPEL provides limited support for dynamic adaptability and the only way to implement runtime changes is to stop a running process, modify the composition, and then restart the process [15]. However, in a volatile environment where modern software systems are expected to operate, it is undesirable and inefficient to terminate and reconfigure a running process every time a change occurs. Moreover, some critical processes cannot be instantly shut down to implement necessary changes. Hence, various BPEL extension efforts adopted mechanisms to create frameworks that enable the dynamic adaptability of processes. Examples of these works, which are called "adaptive BPEL-based frameworks", are discussed in [11]. Such extensions address some dimensions of a dynamic environment such as: changing business rules to fit new requirements, replacing an obsolete partner service, and selecting a new or a better service that becomes available. Consequently, dynamic adaptability is characterised by runtime changes in either the process flow logic or partner services.

In general, the various mechanisms applied to BPEL to realise dynamic adaptability are categorised into four [11]: *(i) message interception*, where adaptability takes effect by intercepting BPEL messages [17]; *(ii) late binding*, where a proxy service is used so that binding to the real service happens at runtime [18]; *(iii) aspect injection*, where aspects are weaved into the BPEL process to accommodate runtime adaptability [15]; and *(iv) explicit integration*, where rule activities are explicitly integrated into the BPEL process [19]. We discuss the details about these various adaptability mechanisms in Section 4.3.

2.3. Structure and binding variabilities

Underlying a service composition is a business workflow, a sequence of business activities driven by business rules. Business rules convey specifications of organisational regulations, policies, and decisions [11]. These rules define and constrain the business process, assert business structure, and influence the behaviour of the business. Considering a composition as a higher level realisation of a business process, its requirements (*i.e.*, software requirements) are subsequently determined from the business rules. The composition's functional requirements specified by the structure and arrangement of component services, are imposed by business rules. Moreover, due to the dynamic nature of organisational and business settings, business rules are often subject to changes. For instance, business rules may take into account the varying service needs of business clients and their circumstances. Hence, this volatility of business rules becomes a primary source of requirement changes.

An adaptive BPEL process can be extended, changed, or customised to adapt to the changing requirements. Additional activities, which associate with external elements like decision rules, service selection rules, composite services, or services that encapsulate rules, are dynamically introduced to the process. We call those activities *variabilities*. A process goes through several options: it may offer new functionalities by adding new activities, it may disable or replace its activities, or it can select or replace its partner services. Moreover, we describe the variabilities as either: (*i*) *structure variability* or (*ii*) *binding variability*. We consider these two variabilities as the most relevant adaptability dimensions for service compositions. On the one hand, structure variability enables a BPEL process to be modified at runtime that leads to a structure-variable process. It introduces the capability of adding, removing, or replacing process activities with the variabilities. This typically describes a variability in the business workflow [20]. For instance, considering the differences in user needs and circumstances, an activity in the workflow may not be requested or a new business rule has to be applied, for a particular user. Thus, some part of the process may be differently performed for each user. On the other hand, binding variability gives the ability of the process to select or replace a partner service at runtime. An ⟨*invoke*⟩ activity in the process may have more than one candidate concrete services that provide the associated functionality. In this case, variability happens when choosing the most appropriate concrete service among the candidates. We regard both variabilities as the basis of our adaptability metrics.

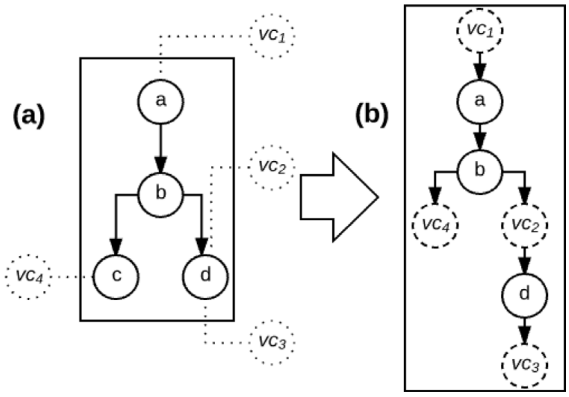

Fig. 1. An execution logic of a process before (a) and after (b) implementing structure variabilities.

We generally characterise variabilities to be activity-driven, *i.e.*, a variability is specified to a particular process activity. We call the place in the process where such variability occurs as a *variation point*. For a structure variability, we consider the *variability concern* $vc \in$ {*Before*, *Around*, *After*} that can be specified to a variation point. We show in Fig. 1 a simplified graph formalism of a process π with nodes that represent activities and directed edges that represent control flow dependencies between activities. We do not consider here data flow dependencies, to simplify the discussion. A directed edge between two nodes means that the activity associated with the origin node has to finish before proceeding to the activity that corresponds to the destination node. Variability concerns vc_1, vc_4, both vc_2 and vc_3 are specified on the variation points *a*, *c*, and *d* respectively. We treat variability concerns differently depending on their placement in a variation point. For example, vc_1 is specified *Before* *a*; vc_4 is specified *Around* (*i.e.*, to replace) *c*; vc_2 and vc_3 are specified *Before* and *After* *d* respectively. Fig. 1(b) shows the execution flow of π after considering the variability concerns. Each vc is associated to a variability element which is independent from the process. Likewise, a vc may be associated with a single activity represented by a single node, or a set of activities represented by a subgraph of nodes. A variability is invoked once the process execution reaches a variation point where a vc is specified. For example, before and after performing *d*, external activities associated to vc_2 and vc_3 respectively must be executed, *e.g.*, to fulfil activities defining new business rules. Likewise, the external activity associated to vc_1 must be invoked before *a*, while the one associated with the *Around* variability concern vc_4 is invoked instead of *c*. The external activities are modified independently upon further changes in the business rules. Hence, the more variability concerns exist, the more likely a process can be modified and adapted to runtime changes. It becomes more plausible that one can find a way to implement external changes when modifying a process. For instance, there are options on how changes are to be accommodated w.r.t. activity *d*. New requirements can be specified or de-specified through vc_2 or vc_3.

For a binding variability, we consider the available *concrete service cs* that can provide the required functionalities of a process. Binding variability has also been studied in [21] and [18]. This variability is particularly associated with the ⟨*invoke*⟩ activities of a process. We show in Fig. 2 a BPEL process in a tree-structure composed of structured activities (nodes n_1 and n_2) and atomic activities (leaf nodes). The process orchestrates three services, s_1, s_2, and s_3, to meet its overall goal. For each service invoked, *e.g.*, s_1, there exist concrete services that match its description, *i.e.*, cs_1, cs_2, and cs_3 are all functionally compliant to s_1. These concrete services may offer similar functionality, but are different in implementation logic and quality of service (QoS). In the figure, we assume cs_2, cs_5, and cs_6 are presently used concrete services. The more concrete services exist, the higher the binding variability for the process. In this case, the likelihood for finding an alternative service that equally provides a needed function gets higher.

3. Quantifying adaptability

In this section we describe our measurement approaches for the structure and binding variabilities. Both start from defining element-level adaptability metrics followed by the metrics for the entire process, derived from these elemental adaptabilities. We further present additional metrics that combine the structure and binding variability, and measure the relative maximal adaptability of a process.

3.1. Measuring structure variability

Our approach aligns with the architectural viewpoint of adaptability [9,22,23], which considers component-level measurement being aggregated to derive a process-level one.

3.1.1. Adaptability of atomic process elements

It is common among adaptive BPEL processes to designate atomic activities as variation points. An activity that can be specified with variabilities is considered *variable*, otherwise *non-variable*. Structure variability is concerned with the variable atomic activities, which we tag each with a structure variability value.

Definition 1. *Structure Variability Value* (*SVV*) — $SVV\langle a\rangle$ is the cardinality of the set of variability concerns $\{vc\langle a\rangle_1, \ldots, vc\langle a\rangle_n\}$ specified to an activity a.

$$SVV\langle a\rangle \mapsto \mathbb{N} \text{ such that } SVV\langle a\rangle = \left|\{vc\langle a\rangle_1, \ldots, vc\langle a\rangle_n\}\right|,$$

where a is a variable atomic activity in the process and $vc\langle a\rangle$ is a specified variability concern for a. □

In Fig. 2, variability concern vc_1 is specified as *Around* to the activity $\langle invoke_{s_3}\rangle$. vc_2 and vc_3 are *Before* variability concerns to $\langle invoke_{s_1}\rangle$ and $\langle invoke_{s_2}\rangle$ respectively; while vc_4 invokes a variability after $\langle invoke_{s_2}\rangle$. To derive the structure variability values, we consider all messaging activities variable, as indicated by the shaded elements. Most adaptive BPEL-based frameworks we have found are able to implement variabilities on the messaging activities. Thus we get: $SVV\langle invoke_{s1}\rangle = 1$, $SVV\langle invoke_{s2}\rangle = 2$, $SVV\langle invoke_{s3}\rangle = 1$, $SVV\langle receive\rangle = 0$, and $SVV\langle reply\rangle = 0$. Both $\langle reply\rangle$ and $\langle receive\rangle$ have no specified vc.

However, absolute values are difficult to interpret especially if being used against other quality attributes. Each value should be mapped into a common range interval, *i.e.*, percentage scale [0, 1], which sets a common interpretation of variability values. To normalise the variability values, a reference value R has to be determined by designers. We therefore set $R\langle a\rangle$ as the reference value for an activity a.

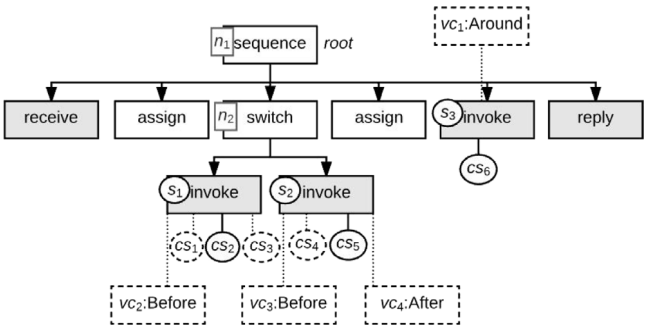

Fig. 2. A BPEL process showing variation points with specified variabilities.

Definition 2. *Structure Variability Degree* (*SVD*) — $SVD\langle a\rangle$ is the *SVV* of a relative to the value of $R\langle a\rangle$.

$$SVD\langle a\rangle \mapsto \{x \in \mathbb{Q} \,|\, 0 \le x \le 1\} \text{ such that } SVD\langle a\rangle = \frac{SVV\langle a\rangle}{R\langle a\rangle},$$

where $SVV\langle a\rangle$ is the *Structure Variability Value* of activity a, and $R\langle a\rangle$ is the maximum number of variability concerns that can be specified for a such that $\forall a$, $SVV\langle a\rangle \le R\langle a\rangle$. □

Referring to Fig. 2, the maximum number of variability concerns for every variable activity is three (*i.e., Before, Around, After*). Based on the structure variability model presented in Section 2.3, $R = 3$ for all a. Thus, we compute: $SVD\langle invoke_{s_1}\rangle = 1/3 = 0.33$, $SVD\langle invoke_{s_2}\rangle = 2/3 = 0.67$, $SVD\langle invoke_{s_3}\rangle = 1/3 = 0.33$, $SVD\langle receive\rangle = 0/3 = 0$, and $SVD\langle reply\rangle = 0/3 = 0$.

We emphasise that the reference value is not arbitrary and has to be precisely determined by designers. In this paper, the value of R, which we use to normalise the *SVV*s, is based on the maximum number of variabilities that we found from the implementations of structure variability among the adaptive BPEL-based frameworks. Cautiously setting this value brings two significance. *First*, the resulting metrics always range in the interval [0, 1] which is commonly interpreted as percentage value. This scale is easy to understand and interpret, which is important for the adoption of any metrics [24]. *Second*, as these element-level adaptability metrics are aggregated to a single metric describing the entire process, such process-level metric can be used for direct comparison of processes.

3.1.2. Deriving process-level adaptability

We define two approaches in aggregating the individual atomic variability degrees in order to derive a metric for the whole process. The first approach is straightforward, deriving the mean of the variability degrees. The second approach, which defines an "effective" variability degree, considers runtime behaviours of the elements constituting the process.

Definition 3. *Mean SVD (MSVD)* — $MSVD_\pi$ is the *SVD* of process π relative to the variable activities of π.

$$MSVD_\pi \mapsto \{x \in \mathbb{Q} \,|\, 0 \le x \le 1\} \text{ such that } MSVD_\pi = \frac{\sum_{i=1}^{n} SVD\langle a_i\rangle}{n},$$

where π is a process having n variable activities $a_1, \ldots, a_n$. $MSVD_\pi$ is the arithmetic mean of the *SVD*s of the variable activities of π. □

We refer to the process in Fig. 2 to compute its *MSVD*.

$$\begin{aligned} &MSVD_\pi \\ &= (SVD\langle invoke_{s1}\rangle + SVD\langle invoke_{s2}\rangle + SVD\langle invoke_{s3}\rangle + \\ &SVD\langle receive\rangle + SVD\langle reply\rangle)/5 \\ &= (0.33 + 0.67 + 0.33 + 0 + 0)/5 = 0.27. \end{aligned}$$

MSVD can give an intuitive measure of adaptability with respect to every variable atomic activity comprising the process.

The higher the value of this metric for a process, the more adaptable the process elements are, on average. For instance, a fully-adaptable process π will generate an $MSVD_\pi = 1$. That is, every variable activity a in π is specified with the maximum variability, which is defined by $R\langle a\rangle$. In contrast, a non-adaptable process π, one without any variability, will have $MSVD_\pi = 0$.

In addition to describing the adaptability of the whole process, *MSVD* can be used to compare different processes. Our aggregation approach, which considers every variable element comprising the process, could allow comparison of various processes regardless of their size. The aggregation of the individual variability degrees is normalised with respect to the number of variable elements in the process. Moreover, *MSVD* will not just provide immediate quantification of the respective variabilities w.r.t. the entire process, but could also imply the degree of complexity of a process in the effort of implementing variabilities. For instance, additional codes that would require some programming efforts are necessary for the inter-operability of variability interfaces with the process. Hence, *MSVD* can offer insight into the mean size of potential interactions of each variable activity with external elements. More interactions have to be managed, as additional variabilities are linked to the process.

MSVD regards equally the adaptability of each variable atomic activity comprising the process. This may not always be the case since atomic activities are integrated within structured activities. Both the execution and the variability behaviour of atomic activities are constrained by the structured ones. The runtime behaviour of structured constructs should be considered in the aggregation of adaptability. Each structured construct has its unique behaviour that affects the execution of activities within it. We argue that by considering the actual execution behaviours of process elements, the metrics become more concrete with a higher chance of accuracy. We define aggregation formulations for the structured activities.

The structured constructs $\langle sequence\rangle$, $\langle flow\rangle$, and $\langle while\rangle$ similarly execute *all* the activities within their scope. Hence,

$SVD\langle c\rangle = \frac{\sum_{i=1}^{m} SVD\langle a_i\rangle}{m}$, where $a_i | i = 1, \ldots, m$ is the ith variable activity within the construct $c \in \{\langle sequence\rangle, \langle flow\rangle, \langle while\rangle\}$, and m is the number of variable activities contained in c.

Furthermore, the conditional constructs $\langle switch\rangle$ and $\langle pick\rangle$ execute only one activity per process instance.

For both constructs, the variability degree depends on the probability of each conditional branch to be executed. The probability is application specific, and can be determined from actual execution data, although methods of determining such probabilities are beyond the scope of this paper. Hence,

$SVD\langle c\rangle = \sum_{i=1}^{m} (SVD_{\langle a_i\rangle} \cdot p_i)$, where m is the number of activities within the construct $c \in \{\langle switch\rangle, \langle pick\rangle\}$; $a_i | i = 1, \ldots, m$ is the ith variable activity; and p_i is the probability of a_i to be executed, $\sum_{i=1}^{m} (p_i) = 1$.

Examples in this paper generally apply a discrete uniform distribution, so that each branch has the same chance of execution, such that $p = \frac{1}{n}$, where n is the number of activities contained in c.

A structured activity can be placed within another structured activity. *SVDs* are hierarchically aggregated to one or more level of structured constructs until a value is derived for the whole process. We define an effective *SVD* of a process applying the preceding formulations in the aggregation of variability degrees.

Definition 4. *Effective SVD (ESVD)* — $ESVD\pi = SVD\langle c\rangle$, where c is the root node in process π.

$ESVD\pi$ is the hierarchical aggregation of *SVDs* considering the runtime behaviour of the structured constructs in a process π. □

We refer to the process π in Fig. 2. We hierarchically derive the $SVD\langle c\rangle$ of each structured construct c until we get the *SVD* of the root node. When a structured construct is an element within another structured construct, the former is considered as a variable activity of the latter.

$$SVD\langle switch_{n2}\rangle = (SVD\langle invoke_{s1}\rangle * \frac{1}{2}) + (SVD\langle invoke_{s2}\rangle * \frac{1}{2}) = (0.33 * 0.5) + (0.67 * 0.5) = 0.5$$

$$SVD\langle sequence_{n1}\rangle = (SVD\langle receive\rangle + SVD\langle switch_{n2}\rangle + SVD\langle invoke_{s3}\rangle + SVD\langle reply\rangle)/4 = 0.21.$$

Since $\langle sequence_{n1}\rangle$ is the root node, $ESVD\pi = SVD\langle sequence_{n1}\rangle = 0.21$.

3.2. Measuring binding variability

Binding variability concerns the $\langle invoke\rangle$ activities in a process. We tag each $\langle invoke\rangle$ activity with a binding variability value. We follow the same approach used for structure variability in Section 3.1, except the mapping of atomic variability values to their corresponding variability degrees.

Definition 5. *Binding Variability Value (BVV)* — $BVV\langle a\rangle$ is the cardinality of identified concrete service choices $\{cs\langle a\rangle_1, \ldots, cs\langle a\rangle_n\}$ specified for an activity a.

$BVV\langle a\rangle \mapsto \mathbb{N}$ such that $BVV\langle a\rangle = |\{cs\langle a\rangle_1, \ldots, cs\langle a\rangle_n\}|$, where a is an $\langle invoke\rangle$ activity in the process and $cs\langle a\rangle$ is an identified concrete service for a. □

In Fig. 2, cs_1, cs_2, and cs_3 are the alternative concrete services for $\langle invoke_{s_1}\rangle$, cs_4 and cs_5 are for $\langle invoke_{s_2}\rangle$, and only one concrete service cs_6 for $\langle invoke_{s_3}\rangle$. We therefore have $BVV\langle invoke_{s_1}\rangle = 3$, $BVV\langle invoke_{s_2}\rangle = 2$, and $BVV\langle invoke_{s_3}\rangle = 1$.

Since there is no determined maximum number of concrete services for an $\langle invoke\rangle$ activity, a reference value cannot be set at once. This is unlike the structure variability where 3 is the resolved reference value for every variable activity. Hence, we skip the normalisation of individual variability values, and use the *BVVs* in the aggregation for deriving a metric of the whole process.

Definition 6. *Mean Binding Variability (MBV)* — MBV_π is the arithmetic mean of the *BVVs* of the $\langle invoke\rangle$ activities of π.

$MBV_\pi \mapsto \{x \in \mathbb{Q} | x \geq 0\}$ such that $MBV_\pi = \frac{\sum_{i=1}^{n} BVV\langle a_i\rangle}{n}$, where π is a process having $n\langle invoke\rangle$ activities $a_1, \ldots, a_n$. □

We refer to the process in Fig. 2 to compute its *MBV*. $MBV_\pi = (BVV\langle invoke_{s1}\rangle + BVV\langle invoke_{s2}\rangle + BVV\langle invoke_{s3}\rangle)/3 = (3 + 2 + 1)/3 = 2$.

With *MBV*, we can compare the binding variability of processes. A higher value derived for the metric means a higher binding variability of the $\langle invoke\rangle$ activities, on average.

Furthermore, considering the runtime behaviour of the structured constructs, we can also compute an effective binding variability following the same way we derive *ESVD*. A similar approach is applied to aggregating *BVVs* within the $\langle sequence\rangle$ and $\langle flow\rangle$ constructs but not with the $\langle while\rangle$. Hence, $BVV\langle c\rangle = \frac{\sum_{i=1}^{m} BVV\langle a_i\rangle}{m}$, where $a_i | i = 1, \ldots, m$ is the ith $\langle invoke\rangle$ activity within the construct $c \in \{\langle sequence\rangle, \langle flow\rangle\}$ and m is the number of $\langle invoke\rangle$ activities contained in c.

It is worth mentioning the difference between the aggregation of *SVDs* and *BVVs* within the $\langle while\rangle$ construct. On the one hand, an activity within the $\langle while\rangle$ construct is iteratively executed, but this does not change its *SVD*. The number of variability concerns, *i.e.*, vc, which is specified to an activity during

design time, does not change. As mentioned in Section 2.3, what only changes at runtime is the external activity associated with the vc. Individually considering the derived *SVD* value for every iteration eventually results to the same value. Hence, the semantics for *SVD* aggregation for the $\langle while \rangle$ construct is similar to $\langle sequence \rangle$ and $\langle flow \rangle$. On the other hand, the availability of concrete services may vary at each iteration $j \mid j = 1$ to n number of iterations. Hence, $BVV\langle while \rangle = \frac{\sum_{j=1}^{n}(BVV'_{\langle while \rangle})_j}{n}$, where $BVV'_{\langle while \rangle} = \frac{\sum_{i=1}^{m} BVV_{\langle a_i \rangle}}{m}$, $a_i \mid i = 1, \ldots, m$ is the ith $\langle invoke \rangle$ activity among the m number of $\langle invoke \rangle$ activities within $\langle while \rangle$.

Similar aggregation applies for the conditional constructs. $BVV\langle c \rangle = (\sum_{i=1}^{m})((BVV_{\langle a_i \rangle} \cdot p_i))$, where m is the number of $\langle invoke \rangle$ activities within the construct $c \in \{\langle switch \rangle, \langle pick \rangle\}$; $a_i \mid i = 1, \ldots, m$ is the ith $\langle invoke \rangle$ activity; and p_i is the probability of a_i to be executed, $\sum_{i=1}^{m}(p_i) = 1$.

Definition 7. *Effective Binding Variability (EBV)* — $EBV\pi = BVV\langle c \rangle$, where c is the root node in process π.

$EBV\pi$ is the hierarchical aggregation of *BVV*s considering the runtime behaviour of the structured constructs in a process π. □

We illustrate this aggregation using the process π in Fig. 2. A structured construct within another structured construct is treated similarly as another $\langle invoke \rangle$ activity.

$$BVV\langle switch_{n2} \rangle = BVV\langle invoke_{s1} \rangle * \frac{1}{2} + BVV\langle invoke_{s2} \rangle * \frac{1}{2}$$
$$= (3 * 0.5) + (2 * 0.5) = 2.50.$$
$$BVV\langle sequence_{n1} \rangle = (BVV\langle switch_{n2} \rangle + BVV\langle invoke_{s3} \rangle)/2 = 1.75.$$

Hence, $EBV\pi = BVV\langle sequence_{n1} \rangle = 1.75$.

3.3. Compound adaptability

Although structure variability and binding variability refer to two different dimensions of adaptability, we can combine them to define a compound adaptability metric for a process. There may be instances when designers need to consider both dimensions in a process evaluation. We apply this combination to the mean and effective calculations of both binding and structure variability.

However, the *MBV* and *EBV* need to be mapped to the common range interval [0,1]. Since the upper limit of these binding variability measures are unbounded, we use a derivative of the sigmoid function in the normalisation. This strictly maps any value of *MBV/EBV* to [0,1], and although sigmoid is a non-linear function, we can approximate the scope for a linear mapping by setting a dispersion factor.

Definition 8. *Mean/Effective Binding Variability Degree (MBVD/EBVD)* — $MBVD\pi/EBVD\pi \mapsto \{x \in \mathbb{Q} | 0 < x \leq 1\}$, respectively refers to the normalised $MBV\pi/EBV\pi$ through the function

$$f(y) = \left(\frac{1}{1 + e^{(-1 \cdot \frac{y}{df})}} \cdot 2 \right) - 1, \text{ where } y = MBV\pi \mid EBV\pi, \text{ and}$$

df is a constant denoting the dispersion factor. □

The dispersion factor has to be carefully determined by designers. For instance in this paper, $df = 10$ approximates a linear mapping for values from 0 to 10, and it provides a gradual scaling with a more dispersed distribution for higher *MBV/EBV* values up to a few tens. Without this dispersion factor, the sigmoid function would quickly map to 1 the values of *MBV/EBV* (*i.e.*, the values above 3 would yield variability degrees close to 1 at a dense interval such that their differences become hard to distinguish), and the subsequent rounding-off would make the results practically similar.

For the process in Fig. 2, we compute $MBVD\pi = 0.100$ and $EBVD\pi = 0.087$.

Definition 9. *Compound Adaptability Metric (CAM)* — is distinguished into $MCAM\pi$ or $ECAM\pi$. $MCAM\pi$ is the weighted sum of $MSVD\pi$ and $MBVD\pi$, while $ECAM\pi$ is the weighted sum of $ESVD\pi$ and $EBVD\pi$.

$$MCAM\pi = (w_x \cdot MSVD\pi) + (w_y \cdot MBVD\pi),$$
$$ECAM\pi = (w_x \cdot ESVD\pi) + (w_y \cdot EBVD\pi)$$

where w_x, w_y are the assigned weights such that $w_x + w_y = 1$, $0 \leq w_x \leq 1$, $0 \leq w_y \leq 1$. □

$MCAM\pi$ or $ECAM\pi$ respectively refers to the compounded mean or effective variability metrics. Both consider at once the structure and binding variabilities of a process π. Weights can be assigned to specify the relative importance associated with each adaptability dimension in the metrics.

Referring to the example in Fig. 2, we assign weights 0.7 and 0.3 to *MSVD* and *MBVD* respectively. Thus, $MCAM\pi = (w_x \cdot MSVD\pi) + (w_y \cdot MBVD\pi) = (0.7 * 0.27) + (0.3 * 0.100) = 0.219$. A similar weighting scheme applied to *ESVD* and *EBVD* derives $ECAM\pi = 0.173$.

3.4. Maximal adaptability

It may be necessary to know the maximum structure variability that a process can obtain considering every element comprising its specification. This primarily regards *all* atomic elements of the process, particularly looking at the proportion of variable elements against the non-variable ones. Although we can similarly augment both *MSVD* and *ESVD* to meet this purpose, we choose the more practical metric *ESVD* as baseline to define a relative maximal structure variability degree.

Definition 10. *Relative Maximal Structure Variability (RMSV)* — $RMSV_\pi = SVD^*\langle c \rangle$, where c is the root node of process π.

$RMSV_\pi$ is the maximal *ESVD* that can be achieved by a process π considering its variable components relative to the non-variable ones. Each atomic activity a is tagged with its maximal structure variability value SVV^*: if a is variable, $SVV^*\langle a \rangle = R^*\langle a \rangle$, where $R^*\langle a \rangle$ is the maximum number of variability concerns that can be specified to a. Otherwise, if a is non-variable, $SVV^*\langle a \rangle = 0$. The maximal structure variability degree $SVD^*\langle a \rangle = \frac{SVV^*\langle a \rangle}{R^*\langle a \rangle}$. For every aggregation $SVD^*\langle c \rangle$, where c is a structured construct, all the elements within c are considered. □

Obtaining a maximal structure variability degree $SVD^*\langle a \rangle$ and aggregating such individual variability degrees follow the *SVD* derivation and aggregation previously described in Sections 3.1.1 and 3.1.2 respectively. We refer to the process in Fig. 2. We maintain our assumptions that all messaging activities are variable and the reference value $R\langle a \rangle = 3$, for all a. As the reference value $R\langle a \rangle$ also implies the maximum number of variabilities for a certain variation point a, $R^*\langle a \rangle = R\langle a \rangle$. Thus, $R^*\langle a \rangle = 3$. As a result, the individual $SVD^*\langle a \rangle$ of the process elements can be either 1 or 0, when a is variable or non-variable respectively. To derive the $RMSV_\pi$:

$$RMSV_\pi = SVD^*\langle n_1 \rangle = (SVD^*\langle receive \rangle + SVD^*\langle assign \rangle + SVD^*\langle n_2 \rangle + SVD^*\langle assign \rangle + SVD^*\langle invoke_{s1} \rangle + SVD^*\langle reply \rangle)/6$$
$$= (1 + 0 + 1 + 0 + 1 + 1)/6 = 0.67, \text{ where}$$

$$SVD^*\langle n_2\rangle = SVD^*\langle switch_{n2}\rangle = (SVD^*\langle invoke_{s3}\rangle \cdot \frac{1}{2}) +$$
$$(SVD^*\langle invoke_{s3}\rangle \cdot \frac{1}{2}) = (1 * 0.5) + (1 * 0.5) = 1.$$

4. Case study: Application of the metrics in the adaptive service compositions and frameworks

We conduct a case study to evaluate the applicability of our metrics and address the following questions:
— *First*, how do we compare the adaptability of processes that may differ in size or specification framework? With the diversity of adaptive BPEL-based frameworks, we investigate the broader usability of our metrics, particularly in comparing processes specified even from different frameworks.
— *Second*, can we quantify the adaptability of specification frameworks? We examine the usability of our metrics by directly comparing frameworks instead of processes.
— *Third*, how does adaptability measurement impact design decisions? Along with presenting the results of our case study, we provide insights that would rationalise the significance of measuring adaptability.

In this section we present the travel booking process used in the case study, the application of binding variability measurement, and our exploration of various specification frameworks for structure variability measurement.

4.1. The travel booking process

Our case study utilises variants of the popular travel booking process, an example frequently used in the literature, e.g., [11,25,26]. The composition logic of the travel booking process, as shown in Fig. 3(a), is expressed in Business Process Modelling Notation (BPMN) [27]. The process is composed of m tasks $t_i|i = 1, \ldots, m$. It starts upon a customer's request for travel booking. The request contains the customer's personal details and travel preferences. Then, it is checked for validity *i.e.*, if the request has correct dates, departure, or arrival cities (t_1). If invalid, the process invokes a Web service to inform the customer about the booking failure (t_2). Otherwise, based on the specified destination in the travel request, the process will book a domestic or international flight (t_3, t_4). Likewise, the process books accommodation (t_5) and rents a car (t_6). The total price gets calculated (t_7) and the best offer with the detailed travel plan is sent to the customer (t_8).

Fig. 3(b) shows a transformation of the BPMN into a BPEL process visualised as an activity tree. This will serve as the generic base model in the case study. It is simplified to include only the required atomic and structured BPEL activities associated with the BPMN process. The atomic activities are the leaf nodes which can be distinguished as variable or non-variable. The structured activities are the internal nodes tagged with the identifiers $n_1, \ldots, n_6$. We also show the assumed partner concrete services $cs_1, \ldots, cs_{16}$ that can provide the functional capabilities for the m number of services $s_i|i = 1, \ldots, m$ orchestrated by the process. The concrete service with solid borderline is presently bound to provide s_i. We denote that task t_i in the BPMN process is fulfilled by service s_i in the BPEL process.

4.2. On measuring binding variability

Measuring binding variability is straightforward in any process regardless of the process size or specification framework. For instance, we get $MBV = 2.0$ and $EBV = 1.844$ for the process in Fig. 3(b), thus, we can assess the present form of the process in terms of its current alternative concrete services. $MBV = 2.0$ implies the average number of identified concrete services per $\langle invoke\rangle$ activity in the process. This can become useful to designers particularly in handling trade-offs between binding variability and other measurable qualities. For instance, increasing binding variability could increase process complexity, as demonstrated in [21]. That is, additional adaptability implementation codes are necessary to create inter-operability interfaces between the concrete services and the process. Hence, designers may set an arbitrary reference variability value during process configuration, *e.g.*, $MBV = 3.0$. Assessing binding variability may lead to either increasing alternative concrete services to improve adaptability, if MBV falls significantly short of the reference value; or otherwise, pruning alternatives that would just add to the composition's complexity, *e.g.*, by removing those alternatives with poorer QoS.

Meanwhile, $EBV = 1.844$ considers not just the number of alternatives, but also the runtime behaviours of the associated structured constructs. EBV provides the expected adaptability, particularly when the process is executed. For instance, the assumed uniform probability distribution for the $\langle switch\rangle$ construct can change in every application. The actual probability distribution, which can be determined from the application's execution history, may change the derived EBV. Referring to the node n_2 in Fig. 3(a), if branch n_3 has a high probability of execution, changes in the number of partner services under n_3 will significantly impact the adaptability. In contrast, a branch with low probability of execution gives less impact to the adaptability. Hence, EBV can give a more precise adaptability measure when the execution domain is known.

4.3. On measuring structure variability

Our case study, however, concentrates more on the measurement of structure variability. The dynamic context of the travel booking process, as a result of constant changes in business environment, organisational policies, and user preferences, increases the tendency of requirements to change. As a result, the business workflow has to change to cope with the new requirements. We assume the following future (new) requirements RQ anticipated by the designer: RQ_1 — the system must verify the identity of the customer upon receiving the travel booking request; RQ_2 — the domestic and international flight booking are merged into one service; RQ_3 — the renting of car is based on the customer's preference, *e.g.*, if the preferred tourist attraction is far from the booked accommodation; RQ_4 — new promotion discount rules are formulated, such as frequent flyers are given 15% discount; and RQ_5 — in addition to the existing notification system, customers are notified by call or SMS depending on their present situation. These RQs represent the potential structure variabilities (vc) that the travel booking process has to accommodate.

Generally, we perform the following steps to evaluate our metrics: (1) transforming the business process into its BPEL form applying the described mechanism of an adaptive BPEL framework; (2) examining how a framework implements process adaptability using the sample set of potential structure variabilities; and (3) measuring the adaptability of the resulting process using the proposed metrics.

We survey various adaptive BPEL frameworks and examine how they implement dynamic changes in the business workflow. We briefly mention in Section 2.2 the categories of adaptability mechanisms applied by these frameworks. For each category, we choose at least one representative framework and examine its approach that enables the given travel booking process to become dynamically adaptable. We summarise the examined frameworks in Table 1 and we provide the implementation details in the following discussions.

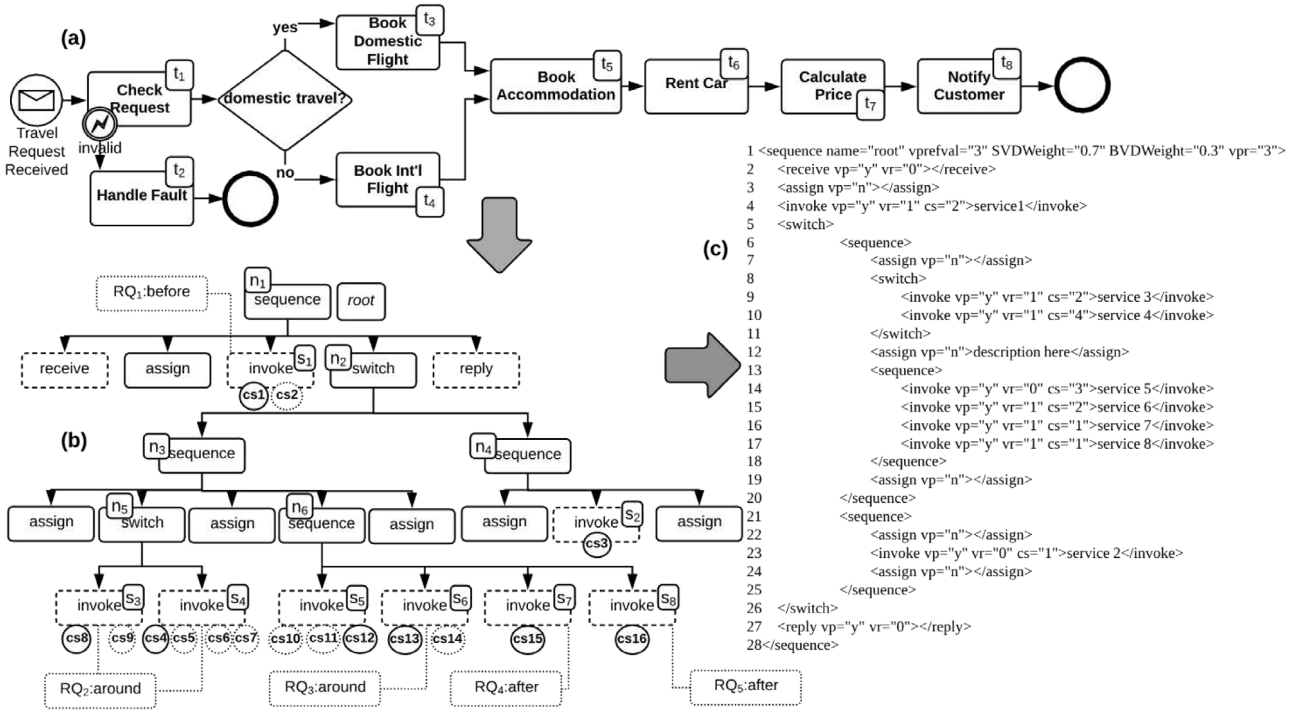

Fig. 3. A travel booking process in BPMN (a) transformed into a visualised BPEL process (b), and its abstract specification (c).

Table 1
The variabilities and variation points of the adaptive BPEL frameworks.

	Adaptability mechanism	Variable constructs	Variabilities per variation point	Supports changes to process flow
AO4BPEL [15]	Aspect injection	<invoke>, <receive>, <reply>	3 (Before, Around, After)	yes
MODAR [11]	Aspect injection	<sequence>	3 (Before, Around, After)	yes
BPELnAspects [16]	Aspect injection	<invoke>	3 (Before, Around, After)	yes
Rosenberg and Dustdar [17]	Message interception	<invoke>, <receive>, <reply>	2 (Before, After)	yes
BPEL+Rules [19]	Explicit integration	<invoke>, <receive>, <reply>	1 (Before)	yes
SCENE [18]	Late binding	<invoke>	1 (Around)	no

4.3.1. Aspect injection

Aspect injection exploits the aspect-oriented mechanism to implement runtime changes to a process. The changes are modularised as separate entities, *i.e.*, aspects, which are integrated into a running process without interrupting its execution. Integrating the aspect-oriented mechanism to BPEL aims to efficiently handle runtime adaptation to new business requirements. These requirements, *a.k.a.* crosscutting concerns, are classified into non-functional crosscutting concerns (*e.g.*, auditing, authentication, logging) and functional crosscutting concerns (*e.g.*, changing business rules, policies, and regulations which affect process flow logic) [15]. We examine three works on aspect-oriented BPEL. These works have similarly utilised the definition of aspects for specifying crosscutting concerns to achieve dynamic adaptability.

(A) *AO4BPEL* [15] considers new requirements as crosscutting concerns. Each requirement is encapsulated into an aspect that defines both the advice, *i.e.*, BPEL activities to fulfil the new requirements, and the pointcut, *i.e.*, specified join point in the process to insert the advice. The aspect also defines the advice type that tells whether to execute the advice *before*, *after*, or *instead of* a join point. AO4BPEL uses activity-driven join points, which means join points are specified on the occurrences of process activities. The implementation of AO4BPEL supports join points on messaging activities particularly on the ⟨*invoke*⟩, ⟨*receive*⟩, and ⟨*reply*⟩ activities. We consider these activities as variable. For each messaging activity, three variabilities can be specified as *Before*, *Around*, and *After* aspects. Fig. 3(b) illustrates an AO4BPEL's implementation of the travel booking process showing the variable activities with dashed borderlines. We also show five aspects that represent the new requirements to be specified in the process. The pointcuts of the aspects, represented by the dotted lines, are counted as the variability concerns for each activity. We illustrate that requirement RQ_2 can be implemented in the process as an *Around* aspect to both s_3 and s_4.

(B) *MODAR* [11] adopts a model-driven aspect-oriented approach to support the development of adaptive BPEL-based systems. The approach has to define a *base model* and a *variable model* for a process. On the one hand, the base model abstracts the main procedure or flow logic of the process. It describes the activities, *i.e.*, general requirements of a business process, and connects these activities using sequential or parallel flows. The base model re-uses BPMN constructs and has two key elements: the *flow objects* are the processing elements, and the *connecting objects* specify the flow relations between flow objects. On the other hand, the variable model captures the volatile decision features of a business requirement. It abstracts into a set of *business rules* the conditional branches in the base procedure (*e.g.*, decision gateway) and the expected changes to requirements specifications (*e.g.*, new business policy). In addition, MODAR defines a *weave model* to associate the elements of the variable model to the base model. Sets of aspects are created such that each aspect *A*, weaves a rule set from the variable model into a business activity in the base model: $A \in \{\{Before, Around, After\} \times T \times RS\}$, where T is the set of business activities and RS is the set of rule sets. The aspects, which can be later invoked as *Before*,

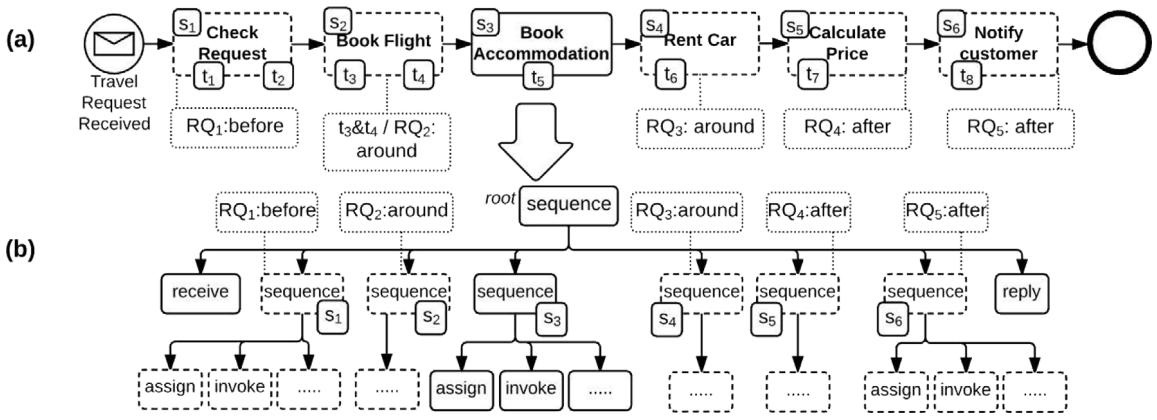

Fig. 4. The MODAR approach: a weave model (a) transformed into a BPEL process (b).

Around, or *After* aspect services, define the corresponding rules for each activity. The weave model is later transformed into a BPEL process, *i.e.*, BPEL code is automatically generated from the model. Fig. 4 shows a schematic BPEL transformation of our travel booking process weave model. A dashed borderline in the weave model in Fig. 4(a), indicates a variable business activity. It is anticipated that the logic of a variable activity is changeable at runtime, through modifying the specified business rule *RQ*. Notice also how the fault throwing in t_1 and the fault handling in t_2, are combined into a variable activity *Check Request*. The rules on decisions with regards to fault management are abstracted in RQ_1. Likewise, the decision gateway from the BPMN model and the attached activities (*i.e.*, t_3 and t_4) are encapsulated to the *Book Flight* activity where the associated variabilities are formed as RQ_2.

In transforming a weave model into its BPEL process equivalent, MODAR uses the ⟨*sequence*⟩ construct to encapsulate each set of BPEL constructs derived from a weave model activity. This abstracts the transformed BPEL process into a series of ⟨*sequence*⟩ constructs. The variation points and variabilities are carried over from the weave model. We illustrate this approach in Fig. 4(b). The variation points are specified to the ⟨*sequence*⟩ structured constructs. In this case, an *SVV* can be directly derived from a structured activity. Regardless of the activities within it, we regard an encapsulation as a single activity which can be a counterpart to the variable atomic construct ⟨*invoke*⟩. In addition, we still maintain that the messaging activities ⟨*receive*⟩ and ⟨*reply*⟩ are variable.

C. BPEL'n'ASPECTS [16] extends the BPEL engine with a notification framework so that the event life cycles of process activities are published and propagated to a broker. The broker handles the aspect management tool (*i.e.*, used to create, edit, delete, deploy or undeploy aspects), the weaver, and the communication between the BPEL engine and Web services. An aspect, which defines the adaptation logic, contains an event subscription (*i.e.*, pointcut) and the advice (*i.e.*, Web service operation). Only an ⟨*invoke*⟩ activity is considered as potential join point. The weaver calls the weaved advice *Before*, *Instead*, or *After* a process activity upon notification from the engine of an event defined by a pointcut in the aspect. This framework can produce a process specification for the travel booking similar to that in Fig. 3(b).

4.3.2. Message interception

This approach implements runtime changes by intercepting messages that go in or out the BPEL process. Generally, the changes are exposed as rule services that can be invoked before or after the execution of a process activity. Among several efforts similarly implementing this approach, we choose to examine the work of Rosenberg and Dustdar [17]. Adaptability is realised by specifying business rules upon intercepting messages exchanged between the BPEL process messaging activities (*i.e.*, ⟨*invoke*⟩, ⟨*reply*⟩, and ⟨*receive*⟩) and the partner services. The runtime architecture components, which include the BPEL engine, the Rule Interceptor, the Rules Broker, and the Web Service Gateway, are connected to the Enterprise Service Bus (ESB). The BPEL engine running the process uses the ESB as a messaging layer, and communicates directly with the Web Service Gateway to call external services. The Rule Interceptor checks every message that goes in or out the BPEL process, and determines from the activity-to-rule mapping document whether a rule is associated to the message. The Rules Broker automatically generates Web services for executing rules, which are invoked *before* or *after* the execution of a partner service. Hence, the messaging activities of a BPEL process are variable. However, only two variabilities: *Before* and *After*, can be specified to these variability points. Considering our travel booking process in Fig. 3(b), the *Around* rules RQ_2 and RQ_3 cannot be applied. This leaves only RQ_1, RQ_4, and RQ_5 as potential structure variabilities.

4.3.3. Explicit integration

In this approach, the standard BPEL is extended by explicitly defining and integrating rule activities in the BPEL process. Rule activities become additional constructs to the existing BPEL elements. A rule activity invokes a rule service that defines new policies and requirements. Implementing this approach is the BPEL+Rules framework proposed in [19]. This introduces a rule-based business process execution environment where a rule engine is deployed on an Enterprise Service Bus and exposed as a Web service. An added *Rule Activity* in the BPEL process invokes a rule service which runs a rule engine and executes a rule logic. The rule service defines a declarative specification of a new requirement arising from a change in business policy or regulation. This supports dynamic adaptability, in a way that rules are modified and applied, without redeploying the BPEL process.

To apply our metrics for this framework, we consider the Rule Activities as variation points of the process. A rule activity is inserted adjacent and prior to any messaging activity in the process. Hence, each messaging activity can be associated to one rule activity as a *Before* variability. Referring to the travel booking process in Fig. 3(b) and the given set of potential variabilities, only the *Before* requirement RQ_1 can be accommodated.

4.3.4. Late binding

This approach intends to provide dynamic adaptability by using rules for the selection of a concrete service that provides a

functionality required by the process. It is assumed that each process instance would require a different partner concrete service such that a service is based on the process execution environment, user, or other contexts. In addition, with several services offering similar functionality, late binding solutions enable the runtime selection of the most suitable service (e.g., having the best quality attribute) or runtime replacement of a misbehaving service.

One framework that demonstrates this approach is SCENE [18]. It extends the standard BPEL by providing rules to guide the binding of a process to partner services. The rules define the constraints, policies, and preferences in the selection or changing of services at runtime. It uses a proxy service that is bound to an ⟨*invoke*⟩ activity. All ⟨*invoke*⟩ activities having the rule-enabled tag set to true, are bound to specifically instantiated proxies. Hence, a designer can leave the concrete partner service of an ⟨*invoke*⟩ activity undefined. At runtime, when the execution of the process reaches the ⟨*invoke*⟩ activity, the operation offered by the proxy bound to that activity is called. The proxy enables the activation of the binding rules that are defined in a rule engine. The dynamic adaptability here is not totally changing the composition logic but by selecting/changing concrete partner services. This capability is similar to the semantics of an *Around* variability, thus, it can still be aligned with structure variability. However, with respect to the given set of potential structure variabilities of the travel booking process, this framework cannot accommodate such assumptions. We rather formulate new assumptions that appropriately apply to this framework. Referring to Fig. 3(b), we assume the ⟨*invoke*⟩ activities, with more than one partner services, are tagged with *Around* variability. Technically, their rule-enabled tags are set to true and yet to be bound to a concrete service. This is in contrast to a non-variable invoke activity which is already bounded to a concrete service, *e.g.*, s_2.

5. Discussions

In this section, we discuss the use of the metrics in comparing processes and frameworks. We also discuss the integration of adaptability measures in performing design decisions, and the limitations to our approach.

5.1. Comparing processes

The main motivation of our metrics is enabling the comparison of processes regardless of size and specification framework, which became possible for two reasons. First, our metrics rely on the standard BPEL constructs and their behaviours. Despite the differences in the adaptability mechanisms implemented in the various frameworks, we observed that the original BPEL language elements are rarely changed. That raises the applicability of our metrics to generally all processes specified from any BPEL-based framework. The examined frameworks in Table 1, except MODAR, can basically produce a specification for the travel booking process similar to that in Fig. 3(b). Second, the aggregation of the atomic adaptabilities into a single value that describes the entire process facilitates direct comparison. Hence, process comparison becomes reasonably straightforward, especially when using binding variability, i.e., *MBV* or *EBV*, where the main concern is identifying the candidate services associated with every ⟨*invoke*⟩ comprising the process.

However, some framework implementation constraints have to be considered when comparing process adaptability based on structure variability. The differences in adaptability implementation among the frameworks of specification are relevant to understand the values derived by our metrics. Table 1 summarises our main observations. *First*, the variable constructs supported by the frameworks are different. These constructs are the process variation points where variabilities can be specified. For instance, AO4BPEL supports variability specification to ⟨*receive*⟩, ⟨*invoke*⟩, and ⟨*reply*⟩, while MODAR specifies variability on a ⟨*sequence*⟩ construct. *Second*, there are differences in the number of variabilities that can be specified in a variation point. Among the frameworks, those implementing aspect injection, allow three variabilities, which are interpreted as *Before*, *Around*, and *After*. Others support only one or two of these variabilities. We emphasise that the number of supported variabilities is the basis of the reference value in the metrics. So far, in the literature, the highest number of variability that can be specified to a variation point is *three*.

Table 2
Across-frameworks adaptability degrees for the travel booking process based on structure variability.

Process specification	$MSVD_\pi$	$ESVD_\pi$	Applicable variabilities
AO4BPEL [15]	0.20	0.12	5 (RQ_1, …, RQ_5)
MODAR [11]	0.21	0.21	5 (RQ_1, …, RQ_5)
BPELnAspects [16]	0.20	0.12	5 (RQ_1, …, RQ_5)
Rosenberg and Dustdar [17]	0.10	0.09	3 (RQ_1, RQ_4, RQ_5)
BPEL+Rules [19]	0.03	0.08	1 (RQ_1)

Furthermore, the designer's choice of the reference value is determined by the *scope of use* of the metrics. We suggest a three-level usage scenario of the metrics in the evaluation and/or comparison of adaptability: (*i*) *across-frameworks* — its use among processes specified from various frameworks, (*ii*) *framework-specific* — for (different) processes specified from the same framework, and (*iii*) *process-specific* — for variants of a particular process. To compare processes specified from a particular framework, we use a reference value that is common among the processes but only becomes valid to such framework. The same case between processes coming from different frameworks, a reference value is determined from considering all concerned frameworks. Meanwhile, a reference value specific to a particular process might produce a more meaningful variability degree for that process. That can be useful when deciding a suitable process configuration. However, by using a process-specific reference value, direct comparison with other processes may no longer apply, thus, depriving the metrics of one of its primary purposes.

In Table 2, we present the results of deriving adaptability degrees for the travel booking process. Given the process and a set of applicable potential variabilities (*RQ*s), we evaluate adaptability w.r.t. the structure variability of its various specifications. This will answer questions such as: which specification of the travel booking process is more adaptable assuming the potential variabilites, or, which process specification can accommodate the future requirements changes defined by the *RQ*s. Only a process specified from frameworks using aspect injection, *i.e.*, the first three rows, can accommodate all the given five variabilities. As expected, these processes have higher variability degrees compared to those with smaller number of applicable variabilities. We do not include in our comparison the SCENE process, which uses the late binding mechanism, because the semantics of the metrics in such specification is different, i.e., the variability of ⟨*invoke*⟩ pertains to selecting a concrete service. We show in Table 1 that SCENE does not support changes in the process workflow. Hence, none among the given future variability assumptions, *i.e.*, *RQ*s, could apply to late binding.

The examined frameworks, except MODAR, do not necessarily change the travel booking process transformation in Fig. 3(b), when implementing adaptability. Hence, we use the same BPEL specification for those frameworks. Looking at the aspect-injection-based frameworks, we get similar adaptability degrees for the processes specified from AO4BPEL and BPELnAspects, as

they both support all the given potential variabilities using the same process specification. Although MODAR also supports all the given variabilities, we get a different result because of its distinct specification of the process. Besides, the MODAR version of the process poses higher adaptability degrees especially for the *ESVD* because of the smaller number of structured constructs involved in the calculation. We show in Fig. 4(b), how MODAR specifies a process and the variabilities, making the aggregation of atomic variability degrees performed at one level of the activity tree. We emphasise that different processes regardless of their sizes and specification frameworks are comparable in an across-frameworks evaluation. The same set of variable constructs has to be considered. That is, in addition to setting a common reference value pertaining to the highest allowed number of variabilities that can be specified to a variable construct. However, when dealing with processes from a particular framework, it may be necessary to have a *framework-specific* evaluation. The metrics become more cognisant of the framework constraints. This produces a more meaningful adaptability degree for a process. The calculation for adaptability becomes more precise with respect to the capabilities allowed by the framework. We take for example the BPELnAspects process in our case study. Despite supporting the specification of a maximum three variabilities per variation point, the BPELnAspects framework allows variabilities to be specified only to the ⟨*invoke*⟩ constructs. If the scope of evaluation is on processes within this framework, it becomes less relevant to include other constructs, *e.g.*, ⟨*receive*⟩ or ⟨*reply*⟩, as variation points to be regarded in the metrics. We refer to the following calculations of the adaptability degrees of the travel booking process with only the ⟨*invoke*⟩ activity as variable: $MSVD_{\pi} = (SVD\langle invoke_{s1}\rangle + \ldots + SVD\langle invoke_{s8}\rangle\ ) / 8 = 0.25$, and $ESVD_{\pi} = SVD\langle n_1\rangle = (SVD\langle invoke_{s1}\rangle + SVD\langle n_2\rangle\ ) / 2 = 0.24$.

As expected, there is an increase in the adaptability degrees from these framework-specific calculations compared to the across-framework measurements derived for the same BPELnAspects process shown in Table 2. If every ⟨*invoke*⟩ activity is specified with three variabilities, we eventually get 1 as the *MSVD* or *ESVD* of the process. A framework-specific evaluation ascertains that the typical value 1 is the maximum adaptability possible for the process. This becomes otherwise, *i.e.*, maximum adaptability < 1, when there are constructs considered variable in the metrics, but are not supported as variation points by the process framework. An ascertained range of process adaptability, *i.e.*, [0,1], facilitates outright interpretation of derived adaptability degrees. For example, a *MSVD* close to 0 means that variable constructs can be much more adaptable, such that additional variabilities can be specified. While a value close to 1 means the process is nearly specified with the maximum allowable variabilities.

Moreover, a framework-specific adaptability evaluation should not only be sensitive to the variation points but also to the variabilities supported by the framework. As we see in Table 1, the frameworks have differences in the number of variabilities allowed per variation point. For instance, frameworks that utilise message interception generally specify a maximum of two variabilities. In this case, when evaluating adaptability of processes specified from such frameworks, we adjust the reference value $R = 2$. Similarly, we can adjust $R = 1$, when evaluating processes particularly specified from frameworks allowing only one variability, such as those we examined that either use explicit integration or late binding mechanisms. This supports our concern in maintaining 1 as the adaptability degree of a process, when all the variation points are specified with maximum variability. Overall, the framework-specific evaluation restricts the scope of the metrics, but the generated results can become more significant in the direct interpretation of adaptability degrees.

5.2. Comparing frameworks

An ideal structure adaptability happens when every element of the process becomes a variation point specified with the maximum allowed variabilities. This means a variability concern of any type can be defined anywhere within the process specification. The process becomes *fully-adaptable* such that $RMSV_{\pi} = 1$. For instance, the process in Fig. 3(b) can be fully-adaptable if variabilities can be specified to ⟨*assign*⟩. However, this may produce significant implementation and operational overheads for the process. For example, specified variability placeholders on places that are not expected to change is unnecessary, and would just add to the complexity that the process has to handle. Likewise, it increases the expected drawbacks to other quality attributes such as performance and response time [9,11]. Such implications may have bound frameworks in their implementation of full-adaptability besides the technical limitations.

The potential level of structure variability that a process may achieve significantly depends on the framework of specification. As observed from the examined frameworks, the adopted adaptability mechanism poses constraints that affect variability. For instance, the differences in either the allowed number of variabilities or the process elements that can become variation points influence the specification's adaptability. It is then important to know how adaptable a certain process can become in a given specification framework. To this aim, we demonstrate the applicability of deriving the maximal adaptability through the *RMSV* metric.

Referring to the travel booking process in Fig. 3(b), we ask the question: what would be its maximal adaptability in a framework utilising late binding such as that of SCENE. Deriving the maximal adaptability considers a framework's constraints in implementing variability. For example, SCENE supports only the specification of an *Around* variability to the ⟨*invoke*⟩ activities. Consequently, only the ⟨*invoke*⟩ activities are variable, and the reference value $R = 1$, which implies $R^*\langle invoke\rangle = 1$. Hence, we derive $RMSV\pi_{(SCENE)} = 0.27$. This derived $RMSV_{\pi}$ reveals the maximum adaptability that the travel booking process π can achieve in its present form in the SCENE framework. The awareness of designers about this maximal adaptability supports a process-specific evaluation that may lead to two ways of improving such potential adaptability degree: first, re-specifying the process in order to either decrease the number of non-variable constructs or increase the number of variable constructs, and second, lessening the depth of the process specification structure. Hence, we can determine a version of a process specification with the best adaptability potential.

Furthermore, the need for an adaptability-intensive composition may require the designer to survey different specification frameworks and select among them the one that can provide the highest level of potential adaptability for a certain process of concern. For instance, given the travel booking process, we can derive its *RMSV* in every framework of specification. However, we need to identify a global reference value $R\langle a\rangle$ to be used across all frameworks. Such value should be the highest R for an activity a, among the different process specifications. Using a global reference value $R\langle a\rangle = 3$, we apply an across-frameworks *RMSV* derivation to the process specifications from the various frameworks and summarise the result in Table 3. Hence, we can now answer questions such as: which framework will provide the highest level of structure variability for the travel booking process, or which framework will enable a process specification to accommodate the most number of variability concerns. Except for MODAR, we use the specification in Fig. 3(b) in the derivation of each *RMSV*, as the frameworks can similarly support such process specification. Accordingly, these frameworks can implement adaptation without necessarily changing that BPEL

Table 3
Relative maximal structure variability of the travel booking process.

Specification framework	$RMSV_\pi$	Maintained baseline specification
AO4BPEL [15]	0.67	yes
MODAR [11]	0.63	no
BPELnAspects [16]	0.27	yes
Rosenberg and Dustdar [17]	0.45	yes
BPEL+Rules [19]	0.22	yes
SCENE [18]	0.09	yes

process specification. A different BPEL specification is used for MODAR. The MODAR framework derives its process specification from transforming its weave model into BPEL, as shown in Fig. 4. Consequently, variable activities are determined from the weave model depending on where the anticipated variability is specified. In this case, a new variability that is yet to be specified somewhere in the process cannot be accommodated. However, MODAR has an *Advanced Adaptability* transformation option that sets all the weave model's variable activities to full-adaptability. That means a predetermined variable activity can be dynamically specified a *Before*, *Around*, or *After* variability. To derive its *RMSV* we utilise the specification illustrated in Fig. 4(b), particularly the top level constructs in the BPEL activity tree. Notice that not all the sequence activities are considered variable, *i.e.*, $\langle sequence_{s3}\rangle$ is not variable. Hence, $RMSV\pi_{(MODAR)} = SVD^*\langle sequence_{root}\rangle = (0+1+1+0+1+1+1+0)/8 = 0.63$.

We emphasise the difference between the Effective Structure Variability Degree $ESVD\pi$ and Relative Maximal Structure Variability degree $RMSV\pi$. Both measure the dynamic adaptability of a process π based on changes in process workflow but have to be interpreted differently. On the one hand, $ESVD\pi$ indicates the level of structure variability of π given the currently specified variabilities. It is assumed that the process designer anticipates the possible runtime changes and specifies a variability placeholder to places where such changes are expected. Adaptability is understood as the adjustments or modifications of such already specified variabilities. Referring to the process in Fig. 3(b), $ESVD\pi$ measures its adaptability based on currently specified RQs, *e.g.*, the adaptability brought by either disabling RQ_1 or modifying the activities defined in RQ_5. On the other hand, $RMSV\pi$ measures the maximum degree of structure variability that a process can achieve in a particular specification framework. Adaptability is interpreted as the capability of the process to be specified with runtime variabilities. For instance, a variability of any type, *i.e.*, *Before*, *Around*, and *After*, is dynamically specified to any variable activity. Referring to the same process in Fig. 3(b), $RMSV\pi$ does not regard the predetermined variabilities. Instead, its concerns are the potential variabilities that the process specification can accommodate, *e.g.*, specifying a new requirement RQ_7: *Around* to the activity $\langle invoke_{s5}\rangle$.

As *RMSV* is sensitive to a framework's capability w.r.t. implementing adaptation for a process specification, this metric can be used to assess not just processes but their frameworks of specification. For instance, the derived *RMSV* results in Table 3 imply that the AO4BPEL framework can offer the most dynamically adaptable specification for the travel booking process in terms of structure variability. Furthermore, *RMSV* can reveal and compare the degree of a framework's *expressiveness* in terms of feature sets. By feature sets, we mean both the supported variability points and variability concerns in a framework. For instance, Table 3 shows that the AO4BPEL and MODAR are more expressive compared to the other frameworks, i.e., both support higher number of feature sets that are necessary to express more desired adaptability of the given process.

Table 4
Impact of adaptability on a quality requirement.

When adaptability increases...	Requirement formulation	
	Higher than	Lower than
The quality attribute value increases	Helps	Hurts
The quality attribute value decreases	Hurts	Helps
The quality attribute is not affected	No effect	No effect

5.3. Considering adaptability in design decisions

The impact of adaptability to the satisfiability of another quality attribute can affect design decisions. We illustrate in Table 4 the typical relationships between adaptability and a quality attribute [9]. The rows indicate three cases for a quality attribute q when adaptability increases: (*i*) q tends to increase; (*ii*) q tends to decrease; or, (*iii*) q is not affected. The columns indicate the formulation of a target requirement pertaining to q, *e.g.*, "availability shall be *higher than* 99%" or "response time shall be lower than 5 seconds". The table indicates that increasing adaptability *helps*, *hurts*, or has *no effect* to the target quality requirement. Generally, when a relationship between adaptability and q exists (*i.e.*, either *case i* or *case ii*), then the alternative process configuration offering the best trade-off is chosen.

We illustrate an example by examining different configurations of the travel booking process shown in Fig. 3b. We observe the relationship between adaptability (*i.e.*, particularly binding variability) and a quality attribute q := "*cost*". We compute the overall q of a process configuration by summing the individual q of concrete services cs composing the process. We assume the following cost values for each cs: {cs_1:40, cs_2:25, cs_3:50, cs_4:25, cs_5:35, cs_6:25, cs_7:30, cs_8:15, cs_9:25, cs_{10}:25, cs_{11}:20, cs_{12}:15, cs_{13}:50, cs_{14}:35, cs_{15}:25, cs_{16}:25}. Table 5 presents the derived adaptability and cost of 10 process configurations where process {a to e}, and {f to g}, are variants of our AO4BPEL and BPEL+Rules specification, respectively. We aim to select the most adaptable process that should satisfy the requirement: "*cost shall be lower than 300 monetary units*".

In Fig. 5, we plot the *EBVD* in relation to cost of the process configurations. Using Table 4, it shows the adaptability and cost relationship belongs to *Hurts*, since when adaptability increases cost also increases, and the cost requirement is formulated as *lower than*. In this case, it is not practical to just select the most adaptable process. Referring to Table 5, process *d*, *e*, *i*, and *j*, satisfy the requirement. We select the one with the highest trade-off score (*i.e.*, $EBVD \cdot cost$), however, process *e* and *j* have equal trade-off scores. Hence, to further reinforce our decision, we use the *ECAM* metric values to compute a trade-off score since we also want to consider structure variability. We note that ECAM provides a single value for binding and structure variability, and we can set the weights of ECAM according to our adaptability priority. We finally determine that process *e* provides the better trade-off score, *i.e.*, $ECAM_A \cdot cost$.

5.4. Limitations

Despite the potential effectiveness of our proposed metrics, we discuss the limitations of our current approach.

Adaptive workflow-based composition frameworks. In our case study, we establish the general applicability of our metrics to the selected representative frameworks. However, we do not guarantee similar effectiveness on adaptability implementations outside the workflow-based composition frameworks (*e.g.*, self-organising adaptability [28]) since our metrics are mainly based on BPEL's workflow-oriented specification constructs.

Table 5
Trade-off between adaptability and cost.

Process instance	*ESVD*	*EBVD*	$ECAM_A$	$ECAM_B$	Cost	Trade-off score (Cost [a] EBVD)
(**a**) Includes all *cs*	0.12	0.092	0.112	0.1	465	42.78
(**b**) w/o cs_8, cs_{13}	0.12	0.087	0.11	0.097	400	34.8
(**c**) w/o cs_6, cs_7, cs_{13}, cs_{10}	0.12	0.083	0.109	0.094	335	27.805
(**d**) w/o cs_2, cs_4, cs_5, cs_6, cs_8, cs_{10}, cs_{11}, cs_{13}	0.12	0.05	0.099	0.071	245	12.25
(**e**) w/o cs_4, cs_5, cs_6, cs_{10}, cs_{11}, cs_{13}	0.12	0.078	0.107	0.091	285	22.23
(**f**) Includes all *cs*	0.083	0.092	0.086	0.089	465	42.78
(**g**) w/o cs_8, cs_{13}	0.083	0.087	0.084	0.086	400	34.8
(**h**) w/o cs_6, cs_7, cs_{13}, cs_{10}	0.083	0.083	0.083	0.083	335	27.805
(**i**) w/o cs_2, cs_4, cs_5, cs_6, cs_8, cs_{10}, cs_{11}, cs_{13}	0.083	0.05	0.073	0.060	245	12.25
(**j**) w/o cs_4, cs_5, cs_6, cs_{10}, cs_{11}, cs_{13}	0.083	0.078	0.082	0.080	285	22.23

[a] $ECAM_A$ uses the weighting 0.7 and 0.3 for *ESVD* and *EBVD* respectively and $ECAM_B$ uses the weighting 0.3 and 0.7 for *ESVD* and *EBVD*.

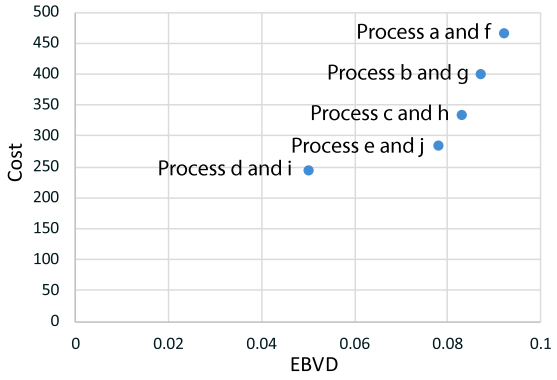

Fig. 5. Relating adaptability and cost.

Equal prioritisation among variation points. Our metrics regard all variation points to have equal importance. However, there are variation points in the process that are vital to the overall operations, hence they should be more adaptable than the less critical ones. The inclusion of variation point priorities in the metrics, might enhance comprehensiveness and accuracy, *e.g.*, a weighted adaptability measurement wherein the weights determine, for instance, "importance" of adapting a process component.

Manual translation of process specification. At present, we manually translate non-BPEL process specifications such as BPMN into the input code specification of our support tool. Since BPMN is the most common process specification language, we need an automated translator of BPMN to BPEL specification. Having such a translator alleviates the burden of encoding, especially involving large processes, and minimises translation errors.

Utilising probability in the variabilities of branching and looping constructs. Determining the actual runtime variabilities can be tricky when involving the branching and looping constructs. In such situations, we presently rely on the designers' knowledge of the application domain and their familiarity with the actual business structure and processes. We deal with the branching constructs by assuming some fixed, application-specific probabilities determined by designers. Likewise with the looping constructs, we assume a predetermined number of concrete services (e.g., the average number of available services from the iterations). Exploiting probability distributions and the Monte Carlo-like simulations [29,30] for those branching and looping constructs, might be beneficial towards a more accurate variability estimation.

6. Automated support tool and comparability evaluation

We have developed a tool AdaptQ, to implement automatic calculation of the adaptability metrics, particularly the aggregated ones describing the entire process. It reads an input XML file that contains the abstract code specification of the process to evaluate. This code abstraction is necessary to uniformly represent the process specifications from various adaptive BPEL frameworks. Each process element is represented, together with its relevant attributes, *i.e.*, the number of associated variabilities and concrete services needed in the calculation. At the moment, we do not support automated translation of process specifications into our code abstraction. Fig. 3(c) shows a snapshot of the travel booking process specification, representing the one specified from the AO4BPEL framework. The attributes within the process element tags indicate the values needed in the adaptability evaluation. Line 1 pertains to the root node where we integrate the reference value $vprefval = 3$ for the *SVD*. Likewise, *SVDWeight* and *BVDWeight* define the associated weights that are needed for computing the combined metrics. This method of assigning varying weights is a design choice and it facilitates the expression of preferences over the represented adaptability dimensions in the combined metrics. The vpr refers to the value associated to R^* when computing SVD^* for the *RMSV*. The boolean attribute $vp = y$ means the element is a variation point, *i.e.*, a variable atomic element w.r.t. structure variability. Otherwise, $vp = n$. The attribute vr and cs respectively indicate the number of associated structure variabilities and concrete services. Specifying these attributes makes our metrics flexible as discussed in Sections 5.1 and 5.2. It enables a designer to set values for the aforementioned attributes according to the identified characterisations of the process, constraints of the implementing framework, weighting preferences, and purpose of the evaluation *i.e.*, across-frameworks, framework-specific, or process-specific.

Furthermore, we investigate how well our metrics perform with regards to comparability, which is an important metric property, denoting an *order* relation between the measurement values [23,31,32].

6.1. Discriminative power

A *discriminative power* [33] describes the ability of the metrics to distinguish between different processes. The more sensitive a metric in differentiating between different processes, the better it can be used for quality comparison and ranking [23]. We explore and compare the discriminative power of the *mean* and *effective* metrics on a given set of process specification variants. We generate the variants through two types of process variations: (*i*) varying the number of variabilities (*resp.* partner services); or, (*ii*) varying the number of variation points (*resp.* ⟨*invoke*⟩ activities).

For the binding variability metrics, we evaluate a total of 192 unique variants of the process in Fig. 3(b). These variants are minimally different with the one in Fig. 3(b) to generate the closest possible differences between the binding variability measures. Firstly, we generate three sets of processes, those with 15, 16, and 17 partner services respectively (the process in Fig. 3(b) has 16). Each set has 32 variants. For every process variant within a set, we use the same number of partner services that are randomly

specified to the ⟨*invoke*⟩ constructs, where each ⟨*invoke*⟩ has at least one partner service. The process structure and its constructs remain unchanged in all variants. Secondly, we generate another three equivalent sets of processes, those with 7, 8, and 9 ⟨*invoke*⟩ activities respectively (the process in Fig. 3(b) has 8). For every variant within a set, we use the same number of ⟨*invoke*⟩ activities that are randomly moved from one structured activity to another. The given number of partner services (*i.e.*, 16), and the process structured activities, remain the same in all variants.

For the structure variability metrics, we also assume a minimal process variation to generate the least possible differences between variability degrees. We evaluate a total of 192 unique variants of the process shown in Fig. 3(b). Firstly, we generate three equivalent sets of processes that are specified with 5, 6, and 7 variabilities respectively (the process in Fig. 3(b) has 6). Each set has 32 variants. For every process variant in each set, the same number of variabilities are randomly specified to the variation points where each variation point has at most 3 variabilities. There are no changes made in the given process structure and constructs. Secondly, we generate another three equivalent sets of processes, those with 9, 10, and 11 variability points respectively (the process in Fig. 3(b) has 10). For every variant within a set, we use the same number of variability points that are randomly placed within the structured activities. The given number of variabilities (*i.e.*, 6), and the process structured activities, remain the same in all variants.

Table 6 shows the descriptive statistics of the computed values per metric. The small values for the standard deviation shows the dense distribution of the derived measures as a result of the slight variations made among the processes. The discriminative power is determined as the percentage of unique values, *i.e.*, each value is rounded-off to three decimal places, derived by a metric from our data set. It can be observed that for both binding and structure variability, the effective adaptability metrics have a significantly higher discriminative power compared to the mean ones. This shows the sensitivity of the effective adaptability metrics to the runtime behaviour of every structured construct. Hence, two process variants even with the same structure and number of variabilities (*resp.* partner services) may differ in their effective adaptability metrics if: (*i*) the variabilities (*resp.* partner services) are specified differently among variation points (*resp.* ⟨*invoke*⟩ activities), and, (*ii*) the variation points (*resp.* ⟨*invoke*⟩ activities) belong to different structured constructs. Meanwhile, the mean adaptability metrics are only able to distinguish the adaptability of processes having different number of variation points (*resp.* ⟨*invoke*⟩ activities) or variabilities (*resp.* partner services).

6.2. Indifference to process size

A well-functioning metric should perform similarly regardless of the process size [23]. Differently sized processes entail differences in both the number of atomic or structured elements and the way the elements are structured. Applying an across-frameworks evaluation, we examine the consistency of our metrics in dealing with these processes. Such consistency justifies the comparison between different processes as discussed in Section 5.1. Hence, we randomly create 36 processes of various sizes and forms by cloning either a portion of, or the whole process specification in Fig. 3(b). The largest process has the size of 134 (*i.e.*, a combined number of atomic and structured elements), while the smallest one has 6. For each process, an assumed number of variabilities or partner services are randomly specified to the variation points or ⟨*invoke*⟩ elements respectively. We derive the metrics for each process as we take note of the relevant process properties such as size, number of variabilities and variation points, and number of partner services and ⟨*invoke*⟩ activities. Utilising our evaluation results from all the processes, we show in Table 7 the correlation coefficient between a metric and a process property. Except for the last column, we do not find significant linear correlations between the metrics and the other four process properties. This means that when individually considering any of these four properties, a smaller or larger value does not necessarily imply a lower or higher adaptability. Moreover, the negative correlation values for the binding variability metrics show that higher measures are likely derived from the ones with smaller sizes in the evaluated processes. Similarly, higher binding variability measures are likely derived from processes with smaller number of partner services, invoke elements, or structured elements. This, however, does not contradict the fundamental notion of binding variability wherein the more partner services exist, the higher the binding variability. It must be likely that the larger processes have smaller overall ratio of partner services to ⟨*invoke*⟩ elements, as compared to the smaller ones.

Contrastingly, the last column shows very strong to maximum correlation values. Both the mean variability metrics *MBV* and *MSVD* have the maximum positive correlation with the mean number of partner services and the mean number of variabilities respectively. This is expected as both metrics are directly derived from the number of variabilities (*resp.* partner services) relative to the number of variation points (*resp.* ⟨*invoke*⟩ activities). Although the derivation of the effective variability metrics *EBV* and *ESVD* strongly relies on the same relationship (*i.e.*, the proportion of variabilities to variation points), there are other factors considered in their derivation, *e.g.*, the structured constructs. This makes the correlation values become less than the maximum. We therefore set forth that the consideration of variabilities (*resp.* partner services) against the number of variation points (*resp.* ⟨*invoke*⟩ elements), enables our metrics to perform consistently even with differently sized processes.

7. Related work

Adaptability of software systems, in particular of service-based systems, has attracted significant attention in the literature, some of which are outlined in [11], and [3]. In general, these works focus on a wide range of concerns such as mechanisms, techniques, and methodologies for the adaptability of service compositions. Our work proposes an approach for the evaluation of adaptability arising from the implementation of such adaptation mechanisms. This evaluation is based on the definition and usage of a set of metrics which can facilitate designers in their design decisions towards a quality composition, leading to the inclusion of adaptability as a first-class quality concern. To put our work in the context of related approaches, we present a survey of the evaluation approaches for software adaptability. In Table 8, we summarise the approaches, including our work, and show the similarities and differences of their characteristics. The common trait among these approaches is that each can be used to evaluate adaptability from either a certain perspective or an adaptability aspect of interest. Some examples are the early works in [34] — where adaptability is based on the proposed measures of software extendability, such as quantifying the number of additions or extensions to an existing system, and in [35] — where adaptability is understood as the degree of maintainability, the proposed measurement scale is based on the time spent in doing adaptive maintenance.

We classify three main adaptability perspectives considered in the approaches: *subjective, architectural, and behavioural*. The subjective perspective relies on user judgement of the adaptability aspect of interest, e.g., [37,38], and [39]. However, there can be limitations of metrics that are based on human judgement

Table 6
Statistics of the evaluated process variants and the derived discriminative power of the metrics.

	No. of variants	Metric	Statistics		
			Mean	Std. deviation	Discriminative power
Binding variability	192	*EBV*	2.146	0.820	0.55
		MBV	2.011	0.165	0.03
Structure variability	192	*ESVD*	0.179	0.104	0.44
		MSVD	0.201	0.022	0.03

Table 7
The correlation coefficient of the metrics with the process properties.

Metric	Process size	Concrete services (*cs*)	<invoke> elements	Structured elements	Mean no. of *cs*
EBV	−0.47	−0.39	−0.49	−0.45	0.99
MBV	−0.51	−0.43	−0.54	−0.49	1.00
		Variabilities (*vr*)	Variation points (*vp*)		Mean no. of *vr*
ESVD	0.13	0.35	0.08	0.12	0.93
MSVD	0.25	0.47	0.20	0.24	1.00

Table 8
Characteristics of the adaptability metrics. (Legend: S-Subjective, A-Architectural, B-Behavioural, AF-Adaptability framework, BP-Business process, GS-General software).

	Adaptability viewpoints			Target of evaluation			No. of metrics defined	
	S	A	B	AF	BP	GS	Multiple	Single
ISO/IEC standard 2001 [36]	✓		✓			✓	✓	
Weibelzahl 2002 [37]	✓		✓			✓	✓	
Gilb 1988 [34]		✓				✓		✓
Fenton 1991 [35]	✓		✓			✓	✓	
Sadat 2004 [38]	✓		✓			✓	✓	
Aldris 2013 [39]	✓		✓			✓	✓	
Raibulet 2009 [40]	✓	✓	✓			✓	✓	
Masciadri 2009 [41]	✓			✓		✓	✓	
Kaddoum 2010 [42]		✓				✓	✓	
Reinecke 2010 [32]			✓			✓		✓
Duenas 1998 [43]		✓				✓		✓
Subramanian 2001 [22]		✓				✓	✓	
Perez-Palacin 2014 [9]		✓				✓	✓	
Lenhard 2014 [24]		✓			✓		✓	
Mirandola 2015 [21]		✓			✓		✓	
Khoshbarforoushha 2016 [44]		✓			✓		✓	
Our work		✓		✓	✓		✓	

w.r.t. reproducibility and reliability. The architectural perspective is concerned with the varying system components and structure, while the behavioural perspective focuses on the adaptability of a particular system property of concern that is observable during system execution, *e.g.,* [32]. Although most works aim to evaluate an adaptability in the general software systems, some works are intended for evaluating the adaptability of specific software domains such as frameworks and processes. Moreover, there are works that propose multiple complementary metrics considering various dimensions of adaptability, *e.g.,* [40,41], and [42]. This is in contrast with works that define a single quantitative metric for evaluating a specific adaptability concern, *e.g.,* [32]. In this section, we elaborate on the existing works that exhibit the aforementioned characteristics and point out specific adaptability concerns addressed by our metrics with respect to these works.

Numerous works have been done in designing software systems prioritising properties such as reliability, performance, security, and maintainability. The styles, analysis, and patterns are well understood for such properties, and standardisations of their evaluation are already in place, *e.g.*, SQuaRE [45]. It has been a different case for adaptability, its incorporation as a primary quality attribute, and how it affects other properties has received less attention. Moreover, although adaptability has long been recognised as a quality characteristic and has been a part of some standardised software quality models, *e.g.*, it is defined as a sub-characteristic of portability in the ISO/IEC standard [36], the perspective may need to vary to accommodate adaptation aspects of higher relevance to service compositions.

Our work, however, is closely related to measures for adaptability based on a software architecture viewpoint. That is considering the system components, their structure, and their interrelationships that will control the overall design and system evolution. In the work of Subramanian [22], software adaptability can be measured in a layered approach by first identifying the adaptability dimension to base the measurement. This dimension refers to any attribute external to the software system, and its changes influence the system, *e.g.*, changes in syntax, character-set, or processing speed. An element adaptability index is assigned to every software component (*i.e.*, 1 for adaptable, 0 otherwise) w.r.t. the defined adaptability dimension. The component indices are aggregated to derive architecture adaptability indices, which are further aggregated to get the overall software adaptability index. This layered approach inspired several other succeeding works such as in [9] and [24]. Perez-Palacin et al. [9] proposed metrics to measure adaptability considering the number of components that provide or can provide the functionalities comprising a software architecture. Moreover, they examined possible relationships between adaptability and other system quality attributes, *e.g.*, availability and cost. If such relationships exist, improving adaptability can cause either improvement or decline on the other attributes. The work of Lenhard [24], which is further expanded in [23], proposed metrics to quantify the degree of structural adaptability of codes written in business process

languages. Lenhard's notion of adaptability is the portability of applications to various runtime platforms. Their purpose is to measure the likelihood of an implementation code to be adapted to another execution platform.

In [21], the authors extended the metrics in [9] particularly for measuring the adaptability of a BPEL process in orchestrating distributed services, based from the number of known services associated with a process element. Moreover, they aim to determine relationships that exist between process adaptability and other quality attributes w.r.t. the different configurations of a process. However, their metrics assume a static process structure that excludes the perspective of often highly dynamic context of processes, i.e., changes in business workflow. Solely relying on this approach to evaluate process adaptability is limited. In particular, processes have been adopting the concept of late binding [18] where the associated services are unknown beforehand. It is possible that a new service that provides a needed functionality becomes available at runtime.

In [46], adaptability of a process is perceived as the flexibility to adjust to new, different, or changing requirements. The authors argue that the degree of coupling of a process to its environment, *e.g.*, partner Web services, clients, and resources, affects the adaptability of a composite service. Accordingly, adaptation should require a lower degree of dependency between a process and its environment. Thus, they proposed metrics to measure the context-independency, *i.e.*, degree of loose-coupling, of a BPEL process. While in [44], metrics are introduced to evaluate a BPEL process on the extent that it can be adapted with future requirements, the authors believing that a process has to be reused in other contexts, organisations, or solutions with minimal effort or change. Although we stand on a similar assumption, that is, the volatility of business rules, the authors focused on quantifying potential mismatches, *i.e.*, description or logic mismatch, that may happen between the process and a potential context of use. Hence, their approach is inclined to predict the reusability of a BPEL process.

Different from the aforementioned approaches, measuring adaptability that is based on the ability of a BPEL process to cope with workflow changes is put forward in [10]. However, such work is confined only to evaluate the variability held by an AO4BPEL process. It defines metrics based on the structure-variability advice types supported in the AO4BPEL framework. It was cut short of examining the applicability to processes specified from other BPEL-based frameworks. Hence, we build on this work and propose an approach to quantify adaptability degree of processes generally defined from the adaptive BPEL-based frameworks. Our approach regards adaptability not just from the perspective of runtime changes in workflow to expose a new process behaviour, but also considers the changes in runtime binding to partner services. These points of view on adaptability are a common focus of the extended-BPEL frameworks, but their corresponding evaluations have been rarely considered in the literature. Moreover, our approach demonstrates flexibility to accommodate a framework specific adaptability evaluation which may be more suitable to some application scenarios.

8. Conclusion and future work

Service composition has become a promising paradigm of fulfilling the demands of modern service-based distributed systems. The volatile contexts of these systems make adaptability a significant property to consider. In this paper, we propose metrics to quantify the adaptability of BPEL processes. We consider two adaptability dimensions: first is the structure variability that changes the execution behaviour of a process, a common adaptation mechanism among extended-BPEL frameworks; second is the binding variability of a process to select a partner service. In the former and latter cases, the metrics respectively reward the availability of more variations in the process execution workflow and the presence of multiple alternatives to existing component services. We demonstrate the applicability of the metrics to processes specified from the different adaptive BPEL-based frameworks. We establish the use of the metrics in the comparison not just of processes but also specification frameworks. Our metrics would facilitate an informed choice among adaptive service compositions and frameworks. We expect the metrics and their potential utilisation as significant contributions towards considering adaptability as a first class concern in the design and implementation of service compositions.

The importance of adaptability and the capability for its measurement support its inclusion as a property to consider in design trade-offs. Along this path, we envision the future directions of this work. There is a need to explore the practical integration of adaptability in multiple quality trade-offs, and evaluation against other service composition concerns. It is also interesting to look for ways to assess the mutual influence between process adaptability and the variability of other quality attributes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] H. Paik, A.L. Lemos, M.C. Barukh, B. Benatallah, A. Natarajan, Web Service Implementation and Composition Techniques, Springer, 2017.
[2] A. Bouguettaya, M. Singh, M. Huhns, Q.Z. Sheng, H. Dong, Q. Yu, A.G. Neiat, S. Mistry, B. Benatallah, B. Medjahed, et al., A service computing manifesto: The next 10 years, Commun. ACM 60 (4) (2017) 64–72.
[3] R. Kazhamiakin, S. Benbernou, L. Baresi, P. Plebani, M. Uhlig, O. Barais, Adaptation of service-based systems, in: Service Research Challenges and Solutions for the Future Internet, Springer, 2010, pp. 117–156.
[4] A. Metzger, K. Pohl, M. Papazoglou, E. Di Nitto, A. Marconi, D. Karastoyanova, S. Agarwal, A. Berre, A. Brogi, A. Bucchiarone, et al., Research challenges on adaptive software and services in the future internet: towards an s-cube research roadmap, in: The 1st International Workshop on European Software Services and Systems Research: Results and Challenges, IEEE Press, 2012, pp. 1–7.
[5] D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Netw. 10 (7) (2012) 1497–1516.
[6] Q.Z. Sheng, X. Qiao, A.V. Vasilakos, C. Szabo, S. Bourne, X. Xu, Web services composition: A decade's overview, Inform. Sci. 280 (2014) 218–238.
[7] I. Mistrik, N. Ali, R. Kazman, J. Grundy, B. Schmerl, Managing Trade-offs in Adaptable Software Architectures, Morgan Kaufmann, 2016.
[8] M. Caporuscio, V. Grassi, M. Marzolla, R. Mirandola, Goprime: A fully decentralized middleware for utility-aware service assembly, IEEE Trans. Softw. Eng. 42 (2) (2016) 136–152.
[9] D. Perez-Palacin, R. Mirandola, J. Merseguer, On the relationships between QoS and software adaptability at the architectural level, J. Syst. Softw. 87 (2014) 1–17.
[10] K.A. Botangen, J. Yu, M. Sheng, Towards measuring the adaptability of an AO4BPEL process, in: Australasian Computer Science Week Multiconference, ACM, 2017.
[11] J. Yu, Q.Z. Sheng, J.K. Swee, J. Han, C. Liu, T.H. Noor, Model-driven development of adaptive web service processes with aspects and rules, J. Comput. System Sci. 81 (3) (2015) 533–552.
[12] C. Pautasso, RESTful Web service composition with BPEL for REST, Data Knowl. Eng. 68 (9) (2009) 851–866.
[13] T. Fleuren, P. Müller, BPEL workflows combining standard OGC web services and grid-enabled OGC web services, in: The 34th Euromicro Conference Software Engineering and Advanced Applications, IEEE, 2008, pp. 337–344.
[14] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, et al., Web services business process execution language version 2.0, OASIS Standard 11 (120) (2007) 5.
[15] A. Charfi, M. Mezini, AO4BPEL: An aspect-oriented extension to BPEL, World Wide Web 10 (3) (2007) 309–344.

[16] D. Karastoyanova, F. Leymann, BPEL'n'Aspects: Adapting service orchestration logic, in: The 7th International Conference on Web Services, IEEE, 2009, pp. 222–229.
[17] F. Rosenberg, S. Dustdar, Business rules integration in BPEL-a service-oriented approach, in: The 7th IEEE International Conference on E-Commerce Technology, IEEE, 2005, pp. 476–479.
[18] M. Colombo, E. Di Nitto, M. Mauri, Scene: A service composition execution environment supporting dynamic changes disciplined through rules, in: International Conference on Service-Oriented Computing, Springer, 2006, pp. 191–202.
[19] A. Paschke, K. Teymourian, C.S.W. AG, Rule based business process execution with BPEL+., in: The 5th International Conference on Semantic Systems, 2009, pp. 588–601.
[20] S.H. Chang, S.D. Kim, A variability modeling method for adaptable services in service-oriented computing, in: The 11th International Software Product Line Conference, IEEE, 2007, pp. 261–268.
[21] R. Mirandola, D. Perez-Palacin, P. Scandurra, M. Brignoli, A. Zonca, Business process adaptability metrics for QoS-based service compositions, in: European Conference on Service-Oriented and Cloud Computing, Springer, 2015, pp. 110–124.
[22] N. Subramanian, L. Chung, Metrics for software adaptability, in: Software Quality Management (SQM), 2001.
[23] J. Lenhard, M. Geiger, G. Wirtz, On the measurement of design-time adaptability for process-based systems, in: Symposium on Service-Oriented System Engineering, IEEE, 2015, pp. 1–10.
[24] J. Lenhard, Towards quantifying the adaptability of executable BPMN processes, in: The 6th Central-European Workshop on Services and their Composition, ZEUS, 2014, pp. 34–41.
[25] A. Charfi, M. Mezini, Hybrid Web service composition: Business processes meet business rules, in: The 2nd International Conference on Service Oriented Computing, ACM, 2004, pp. 30–38.
[26] J. Shen, G. Grossmann, Y. Yang, M. Stumptner, M. Schrefl, T. Reiter, Analysis of business process integration in Web service context, Future Gener. Comput. Syst. 23 (3) (2007) 283–294.
[27] M. von Rosing, S. White, F. Cummins, H. de Man, Business process model and notation-BPMN, in: The Complete Business Process Handbook, Elsevier, 2015, pp. 429–453.
[28] N. Chen, N. Cardozo, S. Clarke, Goal-driven service composition in mobile and pervasive computing, IEEE Trans. Serv. Comput. 11 (1) (2018) 49–62.
[29] A. Brogi, M. Danelutto, D. De Sensi, A. Ibrahim, J. Soldani, M. Torquati, Analysing multiple QoS attributes in parallel design patterns-based applications, Int. J. Parallel Program. 46 (1) (2018) 81–100.
[30] L. Bartoloni, A. Brogi, A. Ibrahim, Probabilistic prediction of the QoS of service orchestrations: a truly compositional approach, in: International Conference on Service-Oriented Computing, Springer, 2014, pp. 378–385.
[31] B. Kitchenham, S.L. Pfleeger, N. Fenton, Towards a framework for software measurement validation, IEEE Trans. Softw. Eng. 21 (12) (1995) 929–944.
[32] P. Reinecke, K. Wolter, A. Van Moorsel, Evaluating the adaptivity of computing systems, Perform. Eval. 67 (8) (2010) 676–693.
[33] H. Zuse, A Framework of Software Measurement, Walter de Gruyter & Co., Berlin, Germany, 1998.
[34] T. Gilb, Principles of Software Engineering Management, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.
[35] N.E. Fenton, Software Metrics: A Rigorous Approach, Chapman & Hall, Ltd., London, UK, 1991.
[36] ISO/IEC, ISO/IEC 9126-1:2001 software engineering - product quality - part 1: quality model, 2001.
[37] S. Weibelzahl, Evaluation of Adaptive Systems (Ph.D. thesis), University of Education Freiburg, Germany, 2002.
[38] H. Sadat, A.A. Ghorbani, On the evaluation of adaptive Web systems, in: Workshop on Web-Based Support Systems, 2004, pp. 127–136.
[39] A. Aldris, A. Nugroho, P. Lago, J. Visser, Measuring the degree of service orientation in proprietary SOA systems, in: The 7th International Symposium on Service-Oriented System Engineering, IEEE, 2013, pp. 233–244.
[40] C. Raibulet, L. Masciadri, Evaluation of dynamic adaptivity through metrics: An achievable target? in: WICSA & European Conference on Software Architecture, IEEE, 2009, pp. 341–344.
[41] L. Masciadri, C. Raibulet, Frameworks for the development of adaptive systems: Evaluation of their adaptability feature through software metrics, in: The 4th International Conference on Software Engineering Advances, IEEE, 2009, pp. 309–312.
[42] E. Kaddoum, C. Raibulet, J.-P. Georgé, G. Picard, M.-P. Gleizes, Criteria for the evaluation of self-* systems, in: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, ACM, 2010, pp. 29–38.
[43] J.C. Dueñas, W.L. de Oliveira, J.A. de la Puente, A software architecture evaluation model, in: F. van der Linden (Ed.), 2nd International ESPRIT ARES Workshop, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 148–157.
[44] A. Khoshkbarforoushha, P. Jamshidi, M.F. Gholami, L. Wang, R. Ranjan, Metrics for BPEL process reusability analysis in a workflow system, IEEE Syst. J. 10 (1) (2016) 36–45.
[45] ISO/IEC, ISO/IEC 25010:2011 systems and software engineering - systems and software quality requirements and evaluation (SQuaRE) - System and software quality models, 2011.
[46] A. Khoshkbarforoushha, P. Jamshidi, A. Nikravesh, S. Khoshnevis, F. Shams, A metric for measuring BPEL process context-independency, in: International Conference on Service Oriented Computing and Applications, IEEE, 2009, pp. 1–8.

Khavee Agustus Botangen received the BS degree in information technology from the Benguet State University, Philippines in 2005 and Master of information systems from the University of the Philippines — Open University in 2012. He is working towards a PhD degree at the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology. His research interests include service computing, recommender systems, and adaptive software engineering.

Jian Yu received the PhD degree in computer software and theory from the Peking University, China in 2002. He is an associate professor at the School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology. He has authored 2 books and more than 90 technical publications. His current research interests include web services and the semantic web, context-aware systems, adaptive software engineering, business process management, and complex networks.

Yanbo Han received the PhD degree in computer science from the Technical University of Berlin, Berlin, Germany. He is a full professor and head of the Cloud Computing Research Centre, North China University of Technology. He has authored more than 160 publications and 5 books. His research interests include distributed systems, business process management, service computing, and large-scale stream data processing.

Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales, Australia in 2006. He is a full professor and head of the Department of Computing, at Macquarie University. His research interests include big data analytics, Internet of Things, and Web science. He was the recipient of the ARC Future Fellowship in 2014, Chris Wallace Award for Outstanding Research Contribution in 2012, and Microsoft Research Fellowship in 2003. He authored more than 350 publications.

Jun Han received the PhD degree in computer science from the University of Queensland, Australia in 1992. He is a professor of software engineering at Swinburne University of Technology. He has authored more than 200 publications. His current research interests include adaptive and context-aware software systems, services and cloud systems engineering, software designs, and software QoS.