SOCA (2013) 7:199–216
DOI 10.1007/s11761-012-0125-z

ORIGINAL RESEARCH PAPER

A systematic literature review of service choreography adaptation

Leonardo A. F. Leite · Gustavo Ansaldi Oliva · Guilherme M. Nogueira · Marco Aurélio Gerosa · Fabio Kon · Dejan S. Milojicic

Received: 26 December 2011 / Revised: 13 October 2012 / Accepted: 6 November 2012 / Published online: 24 November 2012

Abstract A service choreography is a distributed service composition in which services interact without a centralized control. Adequate adaptation strategies are required to face complex and ever-changing business processes, given the collaborative nature of choreographies. Choreographies should also be able to adapt to changes in its non-functional requirements, such as response time, and especially for large-scale choreographies, adaptation strategies need to be automated and scale well. However, the body of knowledge regarding choreography adaptation approaches has not yet been consolidated and systematically evaluated. By means of a systematic literature review, in which we examined seven scientific paper sources, we identified and analyzed the state-of-the-art in choreography adaptation. We found 24 relevant primary studies and grouped them into six categories: model-based, measurement-based, multi-agent-based, formal method-based, semantic reasoning-based, and proxy layer-based. We analyzed (i) how each strategy deals with different types of requirements, (ii) what their required degree of human intervention is, (iii) how the different studies considered scalability, (iv) what implementations are currently available, and (v) which choreography languages are employed. From the selected studies, we extracted key examples of choreography adaptation usage and analyzed the terminology they adopted with respect to dynamic adaptation. We found out that more attention has been devoted to functional requirements and automated adaptation; only one work performs scalability evaluation; and most studies present some sort of implementation and use a specific choreography notation.

Keywords Service choreography · Choreographies adaptation · Choreographies customization · Service composition · Systematic review

L. A. F. Leite (✉) · G. A. Oliva · G. M. Nogueira · M. A. Gerosa · F. Kon
Computer Science Department, University of São Paulo (USP), São Paulo, Brazil
e-mail: leofl@ime.usp.br

G. A. Oliva
e-mail: goliva@ime.usp.br

G. M. Nogueira
e-mail: gmaio@ime.usp.br

M. A. Gerosa
e-mail: gerosa@ime.usp.br

F. Kon
e-mail: kon@ime.usp.br

D. S. Milojicic
HP Labs, Palo Alto, CA, USA
e-mail: dejan.milojicic@hp.com

1 Introduction

Service orchestration refers to a service composition paradigm in which services are controlled in a centralized manner [42]. A single entity known as the *orchestrator* is responsible for managing and coordinating the whole composition execution flow. Orchestrations are described using orchestration languages and executed on an orchestration engine. Orchestration languages explicitly describe the interactions among services by identifying messages, branching logic, and invocation sequences [4]. The Business Process Execution Language (BPEL) [43] is the current *de facto* language for describing orchestrations. A variety of BPEL engines have been developed and made available by both the industry (e.g., IBM Websphere Process Server) and the open-source community (e.g., Apache ODE, JBoss RiftSaw, and OW2 Orchestra). BPEL engines expose orchestrations as ordinary web services.

Service choreography models, in turn, describe multi-party collaboration and focuses on message exchange [4]. Choreography models may be expressed in languages such as WSCI [59], WS-CDL [60], and BPMN2 [44]. Nonetheless, there is not yet a *de facto* standard for choreography modeling. Choreographies also refer to a particular service composition execution model. More specifically, choreographies rely on the peer-to-peer communication style, that is, each service involved in the composition knows exactly when to execute its operations and with whom to interact, without a centralized control [4]. In this sense, choreographies can be seen as an agreement among a set of services as to how a given collaboration should occur. Hence, as opposed to service orchestrations, the control is decentralized.

Since service choreographies do not rely on a single party to coordinate the business processes, some authors claim that they are more scalable than orchestrations [16,42,47,48]. Nanda et al. [42], for example, presents an algorithm to automatically generate a set of distributed processes given a single BPEL process as input and provides experimental results showing improvements on the throughput when services interact without the central coordinator. Liu et al. [33] present an analytical analysis showing that distributed data flows models (choreographies) have better aggregated cost and better response time, given some conditions, than centralized data flows models (orchestrations). Choreographies have also emerged as a promising approach to the future Internet context, in which millions of services, things, and resources take part in large and complex business scenarios [6,25].

Given the pressure to accommodate business changes and the dynamics of infrastructure environments, service choreographies need to be adaptable. In other words, they should be flexible enough as to cope with changes in the *functional requirements*, which usually demand the evolution of the underlying collaboration model (e.g., addition and removal of parties, redefinitions of message exchanges, etc.), and in *non-functional requirements*, such as modifications in the terms of service level agreements (SLAs). Choreographies may also need to be adapted due to violations of predefined quality of service (QoS) threshold values defined in such SLAs, such as response time and throughput. Choreography adaptation is particularly important in large-scale scenarios, where managing and evolving a choreography is even more difficult and complex [45].

Different service choreography adaptation mechanisms exist. Each one occur at a specific time (*design time* vs. *runtime*) and require a certain amount of human intervention (*automated* vs. *manual*). In the service choreography domain, manual adaptation is known as *choreography customization*. Flexible adaptation mechanisms enable software to modify its structure and behavior dynamically in response to changes in its execution environment [35]. According to Nitto et al. [16], adaptation mechanisms for service-oriented applications should be automatic, autonomous, and dynamic. Finally, although any program can be developed to adapt itself, adaptation can also be implemented outside the application, for example, by a middleware system.

Despite the importance of adaptation to choreographies, the body of knowledge regarding the existing adaptation approaches remains to be consolidated and systematically evaluated. In this paper, we present the results of a systematic review aimed at identifying and synthesizing the existing approaches for service choreography adaptation. Given the different kinds of existing approaches, we firstly identified which kind of strategy was employed by each study, grouping them into six categories. After that, we analyzed how each selected study characterized its adaptation strategy according to its (i) support for changes in functional and non-functional requirements, (ii) degree of required human intervention, (iii) impact on the choreography scalability, (iv) implementation, and (v) underlying choreography model. We also selected key examples of each category, illustrating some choreography adaptation uses.

Our contributions include the following: thorough characterization and synthesis of a research area; selection of relevant papers leading toward a taxonomy of adaptation strategies; comprehensive formulation and integration of findings by means of an analysis based on data extracted from the selected papers; and identification of opportunities for future research by discovering missing coverage.

This paper is organized as follows. In Sect. 2, we describe the systematic review protocol, and in Sect. 3, we provide an overview of the selected studies and sources. In Sect. 4, we present the adaptation strategies and their studies. In Sect. 5, we present the discussion, the quality assessment, and possible biases and limitations of this study. In Sect. 6, we describe related work. Finally, in Sect. 7, we state our conclusions, research opportunities, and plans for future work.

2 Research method

Systematic Literature Review is a research method to gather and evaluate the available evidence concerning a specific topic [37]. In contrast to the usual process of literature review, a systematic review reduces bias and follows a precise and rigorous sequence of methodological steps relying on a well-defined protocol. The protocol presents the topic under investigation in a structured question format, as well as instructions for selection, analysis, and summarization of relevant studies. Systematic reviews are thus auditable and replicable, in the sense that other researchers may reproduce the same protocol [17].

Our review protocol was developed and executed according to the guidelines and hints provided by Kitchenham and

Charters [29,37]. The structure of the protocol was adapted from Dybå [17]. We were also inspired and supported by previous systematic reviews conducted in Software Engineering [5,17,26], as well as by previous experiences of our group [54]. A detailed version of our protocol is available at http://ccsl.ime.usp.br/baile/files/systematic_review_protocol.pdf.

2.1 Research questions

As described in the Introduction, our research questions were as follows:

RQ1: What strategy each selected study uses to deal with service choreography adaptation?

RQ2: How each selected study characterizes its adaptation strategy according to the following aspects?

(i) *Target*: Does the adaptation support functional or non-functional requirements changes?

(ii) *Required intervention degree*: Is the adaptation automatically performed? Or is human intervention necessary?

(iii) *Scalability impact*: Is the strategy impact on choreography scalability discussed? Is such discussion informal or does it contain formal proofs/experiments?

(iv) *Implementations*: Is the strategy implemented by a tool or prototype? Is the implementation available for download? If so, is it open-source software?

(v) *Underlying models*: Which choreography models, representations, or standards (WS-CDL, WSCI, BPMN, etc.) are used in the strategy?

2.2 Data sources

We performed automated searches with query string on the following scientific sources: IEEE Xplore, ACM Digital Library, CiteSeerX, SciVerse Scopus, SpringerLink, and Web of Science/ISI Web of Knowledge, which are common sources used in systematic reviews on computer science areas and give a reasonable confidence of covering relevant publications [23]. Additionally, we searched Google Scholar and manually explored the following publications and conferences: JISA (Journal of Internet Services and Applications), IJWSP (International Journal of Web Services Practices), and ZEUS 2011 (Services und ihre Komposition). The manual search did not provide any relevant study according to the established criteria (Sect. 2.4).

2.3 Query string

Searches were performed using a query string considering the papers' titles and abstracts (* stands for the wild card). Two concept groups with synonyms connected with OR operators form the base query: one for choreography and the other for adaptation. Since both concepts must appear on the papers, the groups were linked with an AND operator. Entries containing the plural forms of the concepts (e.g., "choreographies") were included in the search results. The base query, which is adapted to each data source syntax, is the following:

```
  ("choreography" OR
   "decentralized composition" OR
   "decentralized service composition" OR
   "distributed composition" OR
   "distributed service composition" OR
   "decentralized interacting services"
  )
AND
  (custom* OR adapt* OR reconfig* OR
   self-config* OR auto-config* OR "self-healing"
  )
```

2.4 Inclusion and exclusion criteria

To be included, papers should propose or discuss a strategy for choreographies adaptation. Therefore, we have excluded works dealing with the following subjects: simple service selection or adaptation, orchestration adaptation, composition synthesis, and conformance checking. Simple service adaptation occurs when a concrete service implementation is dynamically selected without implying any modification in the interaction among services, as, for example, the work of Cavallaro and Di Nitto [10]. We classified a work as being about orchestration adaptation when adaptation was applied in a composition with clearly one, and only one, coordinator, as occurs in the work of Zeng et al. [65].

When the search returned different papers from the same authors, we verified whether they were about distinct topics. If not, we chose the most complete paper. In addition, to be included, papers should be available for download.

Five graduate students and 3 senior researchers executed the paper selection process according to the stages described in Table 1 (adapted from [29]). In the initial stages, we carried out an inclusive approach, that is, for a paper to be selected, it was considered sufficient if at least one of the researchers suggested it. In the last stage, the result was refined through careful paper examination and discussions until there was an agreement among researchers. We executed the protocol

Table 1 Study selection stages

Stage 1	Researchers applied the search query to all the sources and gathered the results
Stage 2	Researchers excluded duplicated and invalid papers
Stage 3	Researchers applied inclusion/exclusion criteria to the papers titles
Stage 4	Researchers applied inclusion/exclusion criteria to abstracts and conclusions
Stage 5	When necessary, researchers applied inclusion/exclusion criteria to the whole text

in three phases, always with at least two researchers participating. An experienced researcher closely supervised all the process. Two other experienced researchers helped in the protocol definition and in discussing various topics related to this review.

2.5 Data extraction

A structured abstract with the following fields was prepared for each primary study: source; paper title; paper type {journal article, conference paper, short conference paper, workshop paper, technical report, PhD thesis}; authors; year; vehicle; paper abstract; research question/issue; choreography adaption strategy description; human intervention degree {manual, automatic, hybrid}; description of the strategy implementation; strategy limitations and drawbacks; study results/conclusion; study assessment; additional notes.

Besides the structured abstracts, we also used spreadsheets to summarize data regarding publication attributes, answers to the research questions, and studies quality assessment. We have also identified whether the described implementations were available for download, and whether it was released as open-source software.

2.6 Protocol evaluation

As suggested by Kitchenham and Charters [29], before executing the actual systematic review, we evaluated the protocol as follows:

i. Asking five experts in systematic reviews and/or service choreographies to review the protocol. All observations made by the experts were taken into account. Some of their contributions were as follows: additional synonyms for the keywords, advices on how to perform the synthesis of findings, and advices on the systematic review method in general.
ii. Doing a pilot study. We carried out a pilot study of the systematic review on a single search engine, namely IEEE Xplore. By executing the protocol on this source iteratively, we expanded the review scope (manual reconfiguration was included), improved query definition, and improved inclusion/exclusion criteria.

3 Characterization of the selected studies

The execution of the selection process resulted in 24 primary studies on service choreography adaptation. Table 2 shows the number of papers returned by each source, as well as the number of papers that we selected by applying the inclusion/exclusion criteria. The table also presents the differences between sources in terms of included studies (precision) and coverage level (recall).

Table 2 Number of retrieved papers per source

Source	Query results	Selected	Precision (%)	Recall (%)
ACM digital library	55	13	24	54
CiteSeerX	24	3	12	12
IEEE Xplore	54	10	19	42
Google scholar	250	12	5	50
SciVerse Scopus	130	16	12	67
SpringerLink	25	1	4	4
Web of Science	78	7	9	29

Table 3 Paper type distribution

Paper type	Quantity	Percentage %
Book chapter	2	8
Workshop paper	3	12
Short conference paper	3	12
Conference paper	14	58
Journal paper	2	8
Total	24	100

Table 4 Distribution of studies per year

Publication year	Quantity	Percentage (%)
2005	5	21
2006	2	8
2007	4	17
2008	3	12
2009	3	12
2010	4	17
2011	2	8
2012	1	4
Total	24	100

The precision of the sources varied from 4 to 24 % and the recall from 4 to 67 %. SciVerse Scopus was the source with more selected studies (16), but with low precision (12 %). Google Scholar presented the highest number of items returned by the query (250), but the precision was just 5 %. Google Scholar coverage level (50 %) was less than the ACMDL one (54 %) that had just 55 results in the query and the highest precision (24 %). To retrieve all the selected studies, it would be necessary to use at least ACMDL, SciVerse Scopus, and Google Scholar. Using the two sources with highest precision (ACMDL and IEEExplore), the recall would be 46 % (11 selected studies).

We analyzed what kinds of papers were published in the area (Table 3) and how many studies were published per year (Table 4). Searches in all sources were performed on

Table 5 Distribution of studies per focus

Category	Quantity	Percentage (%)
Model-based	8	33
Measurement-based	4	17
Multi-agent systems	4	17
Formal methods	4	17
Semantic reasoning	2	8
Proxy layer	2	8
Total	24	100

June 2012. As sources usually take some time to index new publications, the actual number of publications in 2012 may be higher.

Most papers were published in conferences (58 %), and we found only two journal papers. The distribution per year does not show a clear tendency.

4 Choreography adaptation strategies

In this section, we present synthesized data to answer the research questions. We first show the synthesis by choreography adaptation strategy, which answers **RQ1**, and then, we summarize all the studies per adaptation aspect, according to **RQ2**.

We grouped the selected studies into categories according to their focus: model-based, measurement-based, multi-agent systems, formal methods, semantic reasoning, and proxy layer. The categories give us a perspective about technologies and techniques involved in the area. As presented in Table 5, one-third of the selected studies were grouped into the model-based category and the others are well distributed in the remaining categories.

The next subsections address the systematic review questions by describing each strategy category and the selected studies themselves.

4.1 Model-based approaches

Model-based choreography adaptation strategies are related to model-driven development (MDD) and involve rationalizing about a choreography adaptation at the high-level model instead of the specification and code levels. This strategy is often used to empower business analysts responsible for designing and evolving choreography models, as MDD emphasizes the use of models as the main tool for designing and implementing systems [20]. This strategy also enables automated transformation from one model to another, offering the potential for automatic transforming of high-level abstract models into running systems [36]. A good example of model transformation usage is provided by Mahfouz et al. [34], in which models representing the global choreography, local orchestrations, and business requirements evolve in a coordinated way: Updating one model triggers dynamic adaptation in the other models, avoiding inconsistencies among the models. In total, we grouped nine studies [8,14,15,20,24,34,38,53] into this category.

Fabra et al. [20] explore the idea of domain-experts embodying domain knowledge into the system through high-level specifications, as specialized UML notation. The goal is to enable business processes to support adaptation due to unexpected changes, as earthquakes, without any recoding. Usually, orchestration is described as a sequence of exchanged messages, like in the BPEL language, but this work proposes reaching flexible business process development by separating the business process itself (orchestration aspect) from the service interaction protocol (choreography aspect). According to the authors, the choreography aspect is related to IT infrastructure communication, whose changes are time and money expensive. Such orchestration and choreography views are described in UML extended notations and executed by a Petri net platform in an independent but synchronized way. This architecture enables rapid deployment of business process changes in a SOA-based platform, since changes in the orchestration aspect do not affect the choreography aspect.

Cottenier and Elrad [14] provide a framework for on-demand choreography deployment which reduces conversational and protocol coupling among processes. Conversational coupling is reduced by enabling the initiator partner to choreograph existing reactive processes to build a composite service fitting its requirements. Transactional protocol coupling is reduced by decomposing the collaboration into different layers, each dealing with a specific concern of the application, such as business layer and security layer. This framework is called Executable Choreography Framework (ECF) and introduces a language and platform to enable runtime refinements and composition of processes based on aspect-sensitive processes. These ASPs are process descriptions in which each step is a hook where new actions can be introduced, and each intermediary step is annotated with WSDL. These process descriptions can be refined and composed by third parties, as long as the executable process does not violate the behavioral interface of the ASP. In order to refine the control flow of the choreography, activity diagrams or WS-CDL specifications are translated and partitioned by ECF into several rules, written in an XML-based language called Executable Choreography Language (ECL). Within ECL, several rules specify actions to be performed when a message of interest is intercepted within a target process. ECF-enabled platforms can then interpret ECL rules and deploy the corresponding control and data flow logic, in the native language of the platform.

Dar et al. [15] aim to bridge physical and Internet world by exploring service orchestration and choreography undertaking scalability and dinamicity issues of Internet of Things. They claim that in the context of dynamic IoT environments, where a large number of volatile devices and services are available, adaptability becomes a key features, as it allows an application to continuously change itself to satisfy new contexts. Their research challenges are as follows: (i) awareness and adaptability, which is the ability of the system to provide service composition adaptation and context-aware dynamic services; (ii) scalability and re-configurability, or how to design and dynamically compose services related to huge amount of IoT resources; and (iii) dynamicity. To tackle these, they propose a composition in two levels: local orchestrations and global choreography process. Orchestration is used among devices, as only a small number of them are usually attached to a gateway (or sink), therefore, allowing to avoid peer-to-peer interaction among nodes, reducing the complexity of communication and, therefore, making better use of the limited resources this kind of devices have. A choreography is used for the gateways node, allowing parallelism, and reduced message exchange. This choreography will run in a distributed middleware acting as Service Choreography Engine and will be described by BPMN2 notations that are translated into BPEL4Chor code. To provide runtime reconfiguration, they will use SCA-compliant tools, such as FraSCAti on gateways and Contiki operating system for IoT devices. They plan to implement design time reconfiguration in which the user may customize the composition through a graphical user interface, which will generate the models to be translated and deployed on-the-fly.

Stegaru et al. [53] propose an adaptable and dynamic interoperability high-level model for service choreographies, incorporating QoS parameters. The author considers that adaptability can be mapped to three interoperability levels: organizational, operational, and technical. At the organizational level, adaptability is viewed as responding positively to changes in business process, and user's requirements, while interoperability as dealing with how these requirements are expressed. At the operational level, interoperability refers to interactions between services, and adaptability is concerned with providing alternatives in response to external context change. Finally, at technical level, interoperability is concerned with the format of message exchange among web services. The business layer encompasses the domain knowledge, as well as QoS requirements and interdependencies among business models that need to interoperate. Their model is extremely generic, and as so, they claim that it can be applied to several different use cases.

Moo-Mena and Drira [38] handle repair faults at the architectural level, focusing on service duplication and substitution, and define adaptation rules as graph transformations. The first rule defines the basic actions needed to add a web service to the system. The second rule deals with the addition of connections between web services. The third rule defines the necessary actions to remove elements and any associated connections. Such rules enable developers to understand and to adapt the choreography as a graph model, without the need of understanding how service communication is implemented. The transformation rules could also be invoked by any adaptive software that identifies some evolving requirement.

The study of Mahfouz et al. [34] specifies a derivation algorithm that generates a choreography description based on requirements defined in the Tropos notation, an agent-oriented methodology for software development focused on organizational requirements at various levels of abstraction. Tropos models capture goals of participants in the interaction, mutual dependencies that motivate them to interact, and activities they undertake to achieve their goals, including physical activities. The adaptation algorithm uses the requirements' description to infer constraints on message exchange, which can result in automatically adding, removing, or reordering message exchanges.

Hiel et al. [24] make service orchestrations adaptive to choreography changes based on a model in which developers specify what should be achieved by which partner rather than how this should be reached. Adaptability is enabled through more flexible service compositions, for which developers specify only key information to be exchanged in the choreography. The adaptation is performed by scripts interacting with a manageable orchestrator using three techniques: remapping, reordering, and recomposition. Remapping ensures consistence between objectives and messages. Reordering defines alternative message orders if necessary. Finally, recomposition checks whether there are alternative service configurations that can solve a conformance problem.

Bræk et al. [8] have wrote a book chapter that addresses concepts and methods to support dynamic service compositions. Service interactions are modeled with UML and associated with their executing environment, called situations. The adaptation strategy uses a policy-driven mechanism to react to situations and to select the services to be used within a composition, also considering user preferences.

Although sometimes choreographies and orchestrations are seen as excluding approaches, some works in the model-based category endorse that they are actually different views of service composition, as posed by Poulin [48]. Fabra et al. [20] define orchestrations as the business processes workflows of each party and choreography as the description of the stable interactions among services. They separate and coordinate these aspects in runtime by using two interacting engines: a workflow engine for orchestration and a protocol engine for choreography. Moreover, Hiel et al. [24] define choreography as a contract to which business partners should

adhere and as a (global) perspective to assist developers in creating inter-organizational processes.

Regarding **RQ2**, the studies focusing on model-based approaches cover the following:

Target: All the studies found within this category handled functional requirements (embodied domain knowledge and requirements description). Three studies [8,14,53] dealt also with non-functional requirements.

Required intervention degree: Dar et al. [15] consider both automatic and human intervention, whereas Mahfouz et al. [34] and Moo-Mena and Drira [38] consider only manual adaptation. All the others deal only with automated adaptation.

Scalability impact: Only the work of Dar et al. [15] considers scalability issues. The authors identify scalability as a issue to be tackled in the very large-scale Internet of Things context. They also declare that their system is going to be tested with a huge number of composed resources. However, the solution to the scalability issue is presented as ongoing work, and the authors do not show possible ways to handle it.

Implementations: Two following were found: the choreography customization framework [34], which implements a derivation algorithm; the DENEB [20], an ESB (enterprise service bus) based on Petri nets that executes both choreographies and orchestrations; and the Executable Choreography Framework [14], which uses Aspect-Based Processes and an XML-based language to provide runtime refinement and composition.

Underlying models: identified notations in this category were UML [20,38]; ACDL [34], which is based on WS-CDL; Aspect-Sensitive Process, WS-CDL and the XML-based ECL [14]; BPMN2, BPEL and BPEL4Chor [15]; and a graphical model provided by the OperA tool [24], which enables developers to specify choreographies at a high level of abstraction. It is worth to note that every study in this category has clearly defined an underlying choreography model, except for Stegaru et al. [53].

4.2 Measurement-based approaches

Measurement-based adaptation is triggered when a given threshold is violated by the choreography, and the recomposition is calculated to avoid this violation. This strategy is often used to ensure system invariants, mainly quality of service (QoS) parameters. For example, in Colman et al. [12], library managers are interested in monitoring *terms-of-trade* to buy books from supplier services. Such terms-of-trade include cost, delivery time, supplier reputation, etc., which are QoS parameters for this application. Considering all these variables, a broker service chooses a supplier service. Since different suppliers may present different interaction patterns, the broker is able to dynamically change its interaction behavior to adapt to the chosen supplier service.

An important piece of the measurement-based approach is the "monitor," which holds a global view [19] of the system and is responsible for triggering the adaptation when some invariant is violated or a service fault occurs. Such violations can also be continuously predicted by the monitor [63], which can be achieved by the utility functions usage [46]. The monitor can also play an active role, holding the behavioral and recovery policies that enable the choreography to be reconfigured [18]. To avoid having the monitor as a single point of failure, Ezenwoye et al. [18] state that multiple monitors could be used. In Colman et al. [12], the monitor is referred to as "adaptive application-specific middleware" and is responsible for setting non-functional requirements and measuring QoS. To deal with functional requirements, monitors may track exchanged messages and try to detect unexpected content [19]. A weakness of the measurement-based approach is that retrieving the global state in distributed systems is not a trivial task: a peer state can change just after the old state is sent to the monitor, and synchronizing distributed monitors presents similar challenges. In this category, we counted five studies [12,18,46,63].

Paspallis and Papadopoulos [46] strategy performs continuous evaluation and selection of a composition that maximizes the utility function, which is a quantifiable measure of service quality. The authors suggest two kinds of reasoning: proactive and reactive. Proactive reasoning is more likely to achieve faster and more accurate results and requires all context data to be communicated as soon as it becomes available. In contrast, reactive adaptation reasoning is better in terms of resource consumption, as it defers the communication of such contextual data until they are actually needed. The authors also argue that the proposed strategy provides additional benefits such as robustness, agility, and scalability, but without explicit evaluation of these aspects.

Yang et al. [63] propose a strategy to achieve global optimization in decentralized service composition by periodically predicting QoS. Prediction is based on a semi-Markov probabilistic model, using the "EX-QoS model," which extends the traditional QoS by considering the relation of data transmission between services. The process of reselection, based on a heuristic algorithm, is triggered when a service is predicted to become unavailable. Such prediction strategy does not affect the runtime performance, since when invocation happens, the reselection has been already done. Simulation experiments were performed to test: (i) the effectiveness of the proposed performance predicting approach based on the semi-Markov model; (ii) the performance of the heuristic algorithm for finding replacement; and (iii) the effectiveness of the EX-QoS model. All the experiments resulted in favor of the authors' proposals.

Ezenwoye et al. [18] identify the requirements for dynamic reconfiguration of data-intensive service compositions and poses a reconfiguration strategy based on the monitoring approach, and in a hybrid composition model, combining the positive attributes of orchestration (easy management) and choreography (efficient data flow and scalability). The monitoring is based on a WS-CDL-derived model that represents the global view of the service interaction. WS-CDL is extended by introducing redundancy and other constructions (fault-tolerance pattern). By analyzing the model, the monitor detects failed executions, checks for violations of desired invariants, detects deadlocks, and computes global quantities such as cost. When the monitor detects some of these behaviors, an adaptive action is triggered according to specified recovery policies.

Finally, Colman et al. [12] introduces adaptive runtime role structures that enable services to be composed and autonomously reconfigured. Dynamic contracts control interactions between services by setting and measuring non-functional requirements for these interactions. The paper describes a middleware that can adapt services in two ways: regulation and reconfiguration. Regulation involves setting non-functional requirements in the contracts. Reconfiguration involves changing role structure by creating or destroying roles and contracts, or swapping services. Changes to non-functional requirements, computed as utility function objects, can only be triggered by means of the management interface, available either from another composition or by a supervisory control program.

The category aspects regarding **RQ2** are as follows:

Target: Only Ezenwoy et al. [18] concern functional requirements; two out of five studies concern non-functional requirements [46,63], and only one concerns both requirements at the same time [12].

Required intervention degree: Three studies [12,46,63] were classified as performing automated adaptations, while one study [18] was classified as presenting a hybrid approach.

Scalability impact: Paspallis and Papadopoulos [46] claim to achieve scalability through the use of utility functions; however, being a short paper, scalability is not further addressed. Yang et al. [63] have the only work that performs experimental validation of scalability and do it simulating the dynamic creation of multiple scenarios, varying the amount of service roles and services for each role.

Implementations: The study of Paspallis and Papadopoulos [46] presents an extension of the MADAM middleware, which is a platform to support adaptive behavior in mobile applications. The MADAM middleware source code is available on the project website[1] under the GPL license. Yang et al. [63] do not clearly provide an implementation in their work, but the performed experiments suggest the existence of some. Colman et al. [12] introduce "adaptive application-specific middleware composites," built by using the ROAD framework, which enables services to be composed and autonomously reconfigured.

Underlying models: We have identified no underlying models in this category. However, another paper from Ezenwoye and Tang [19] uses WS-CDL. Such paper is a simplified version from the Ezenwoye's selected study.

4.3 Multi-agent-based approaches

Multi-agent systems (MAS) are systems in which programs called "agents" interact with each other, achieving a set of goals or accomplishing tasks [32]. Each agent has autonomy to use the most appropriate approach to solve its particular problem, and coordination is necessary only when interdependencies arise [56]. Agents can also learn, self-analyze, and adjust their own behavior according to the environment, even with partial knowledge about the whole context. Agents are often implemented with artificial intelligence techniques, such as execution plans [64], and the BDI (beliefs, desires, and intentions) model [40]. In the service choreography context, agents can also be used to recover from interaction failures based on declarative policy specifications [7].

The strategy presented by Yau et al. [64], for example, is based on agents that exchange information to produce and execute mission plans. In this scenario, agents may dynamically invoke mission planners to retrieve execution plans that specify how to coordinate invocations to other services. Such execution plans may be recalculated during their execution if context properties are violated. In the provided example, the execution plan is a rescue plan involving different ships and a helicopter. Each ship and the helicopter are coordinated by different agents. The context property is the number of life-threatening injuries.

Although the multi-agents collaborative behavior seems to be well suited to the distributed coordination paradigm, some authors, such as Fernandez-Llatas et al. [21], claim that multi-agent approaches may suffer from performance issues.

Four studies explicitly use a multi-agent system to dynamically adapt web service choreographies [7,40,62,64].

Xu et al. [62] propose a framework for dynamic service composition that relies on Spi calculus, which is an extension of π-calculus especially designed for the description and analysis of cryptographic protocols [1]. In particular, the framework defines four roles that the agents can play and formally describes how the interaction between such roles takes place using Spi calculus. The framework is then applied by "agentifying" choreography services and making the agent roles exchangeable. Although no detailed empirical evidence is given, the authors claim that such exchange can improve

[1] http://www.intermedia.uio.no/display/madam.

robustness, adaptability, and reliability of web services composition.

Morreale et al. [40] argue that current web service technology does not meet the real business needs. It says that instead of using each other, companies create collaborative relationships, with interactions respecting the identity and autonomy of each party. To support such interactions, web services must be more flexible, focusing on message and conversations instead of invocations. Aiming at providing such flexible system, they define a software component that enables agents to dynamically establish conversations with other agents without any specific interaction protocol. Their interactions are defined by an extended WS-CDL notation, which can be set at design time or runtime, thus allowing agents to take part in conversations previously unknown to them. Such component is included in the PRACTIONIST framework [39], which aims to support the implementation of agent systems according to the BDI model. The framework also provides each agent with a choreography engine, which is able to parse and execute the extended WS-CDL files. The authors state that defining shared choreographies for specific conversations could support the collaborative attitude and the adaptive behavior within agent societies, especially in open and dynamic environments.

Blanchet et al. [7] investigate how to avoid workflow failures that result from out-of-sync distributed workflow models maintained by different organizations. In particular, the authors tackle problems that are detectable through certain types of conversation errors. The authors propose an approach that defines an agent layer, which wraps each web service, operating with a conversation layer that defines normative message exchange based on the underlying workflow model. Deviations from this normative message exchange trigger a conversation error, which is regarded by both agents as a symptom of mismatching workflows. The agents consult policies to resolve which model is to be used as the correct one and then restart their interaction.

Xu et al. [62] propose a model for situation-awareness (SAW) requirements in service-based systems. SAW refers to the capability of being aware of situations and adapting the system's behavior accordingly. In addition, the authors develop agents to incorporate SAW and adaptive coordination in service-based systems. The approach assumes the existence of a mission planner (MP), which is a component that accepts goals specified by the users and generates execution plans based on available services and current situation. In turn, SAW agents are mainly responsible for situation analysis (monitoring) and coordination of services. In summary, SAW agents are able to adaptively coordinate services in execution plans as follows: (i) At each step of execution, SAW agents check whether all the dependencies are satisfied; (ii) if the dependency on a situation is not satisfied, SAW agents check whether the step can be undone; (iii) if the step is undoable, SAW agents first undo the step and then search for an alternative service; (iv) if an alternative service is found, SAW agents resume the execution using the alternative service; otherwise, SAW agents notify MP to do the replanning to find an alternative workflow.

The category aspects regarding **RQ2** are as follows:

Target: The studies of Morreale et al. [40] and Blanchet et al. [7] deal with functional requirements. The studies of Xu et al. [62] and Yau et al. [64] handle both functional and non-functional requirements. Xu et al. [62] are the only authors that consider security issues.

Required intervention degree: All studies of this category deal with automated adaptation.

Scalability impact: No studies in this category showed concerns with scalability.

Implementations: We have found the security-aware CSMWC, based on Spi calculus [62], in which agents retrieve requirements, search for services fulfilling these requirements, and perform the adaptation when necessary; the PRACTIONIST framework,[2] released under the LGPL license and based on BDI techniques [40], in which the message exchange sequence is abstracted to the user, and handled by the agents; and the Workflow Reconfiguration with Agent- and BPEL-Based Intercommunication Technology (WRAB-BIT) [7], which recognizes conversation errors and supports dynamic adjustment of workflow scripts to avoid such errors.

Underlying models: multi-agents adaptation strategies employed WS-CDL [40] and Spi calculus [62] as underlying models.

4.4 Formal method-based approaches

This category encompasses adaptation strategies that rely on either process calculus or finite state machine models. Process calculus is a diverse family of related approaches to formally model concurrent systems. Process calculi support high-level description of interactions, communications, and synchronizations among independent agents or processes [3]. Process calculi also provide algebraic laws that enable process descriptions to be manipulated and analyzed, and enable formal reasoning about equivalences between processes. Regarding service choreography, such techniques are used to check the realizability of a choreography and the automatic generation of prototypes [51].

Leading examples of process calculi include CSP, ACP, LOTOS, π-calculus (pi-calculus), and CCS process algebra. Depending on the process calculus employed, it is also possible to derive finite state machines [50], finite state processes (FSPs), or labeled transitions systems (LTSs) from the formal models [51].

[2] http://www.practionist.org.

Using FSP and LTS representations, Roohi et al. [51] provide peer generation, realizability check, and configuration check to a simple choreography: A broker service buys metal from a market service through an auction system, and a board service publishes the auction result.

Adaptations strategies are usually implemented by means of a formalization of choreography definition or their requirements [27] in a process calculus notation. We found five studies in this category [27,50,51,61].

Roohi et al. [51] propose an encoding of Chor (simplified version of WS-CDL) into the FSP process algebra. From such encoding, the authors investigate whether a set of peers can be generated to realize a choreography. Furthermore, the authors propose a method to check the reconfigurability of a choreography. The reconfiguration check receives as input two choreographies C_I and C_R written in Chor, where C_I stands for the initial choreography and C_R stands for the reconfigured one. A trace, containing the interactions history of a current enactment, is obtained from C_I. If the trace can be reenacted in the peers generated from C_R, then C_R is accepted, which means that the actions that took place before the reconfiguration can be reproduced in the new choreography. Finally, the authors propose a reconfiguration strategy in which actual peers matching abstract descriptions (LTSs) derived from C_R are taken from databases of peers (e.g., UDDI), instantiated, and executed according to the history stored in the trace.

Wombacher [61] claims that business critical information is maintained in the orchestration and not shared with choreography partners. However, changes in the choreography level may require other services to adapt their choreographies and orchestrations, and therefore, there needs to be integration of changes in choreography level into existing orchestrations. The procedure proposed by Wombacher uses annotated finite state automata (aFSA) to represent choreographies and nested word automata (NWA) to represent BPEL orchestrations. Together with a set of axioms and invariants, it is possible to superimpose a semantic representation of the orchestration. The approach is able to propagate: subtractive changes (removing message sequences) and additive changes (adding message sequences), derived using the DYCHOR [50] method, and adapts the resulting NWA to conform with BPEL language. The method outputs a set of change recommendations to be evaluated by the developer, and if approved will generate to a new BPEL.

Jureta et al. [27] state that the openness and adaptability of service-oriented systems influence the way in, and degrees to, which initial functional and non-functional requirements are satisfied at runtime. The authors argue that, to remain relevant after deployment, initial requirements specification ought to be continually updated. The authors call "Client RE" the requirements engineering (RE) devoted to quality parameters and constraints guiding service composition. Client RE assumes a coordination mechanism that is defined, and guided, by constraints to be followed and quality parameters to optimize (e.g., QoS, execution time, service reputation). In order to assist existing RE methods in dealing with Client RE, the authors propose the Dynamic Requirements Adaptation Method (DRAM). DRAM relies on updated rules that automatically (or with limited human involvement) change the initial requirements specification according to quality parameters, their values, and constraints on inputs and outputs characterizing the services composed at runtime to satisfy service requests. The strategy main limitation regards the lack of automated means for defining or facilitating the definition of updated rules. In summary, instead of describing how a choreography should be adapted, the work discusses how an initial choreography specification (along with quality attributes) could be kept updated in the face of runtime changes.

Rinderle et al. [50] present an approach to support the controlled evolution of business process choreographies. Partners involved in a choreography often exchange messages via their public processes, which can be considered as special views of their private processes (orchestrations). Therefore, the authors discuss how changes in a private process may affect the public process of its own enterprise and how they can possibly propagate to other partners. To be able to determine whether a change needs to be adequately propagated, the authors introduce a formal model based on annotated finite state automata (aFSA). It checks whether a change adds message sequences to the public process automation or remove message sequences from it. Then, it indicates whether a change causes effects on partners' processes by checking the consistency between the public processes of two organizations. In order to check whether two public processes are consistent, the two respective aFSAs are intersected. Due to the autonomy of the partners and due to privacy of business decisions, an automatic adaptation on partner's private process is not desired, but the proposed framework assists the user in correctly accomplishing this task by suggesting adaptations.

The category aspects regarding **RQ2** are as follows:

Target: All the studies of this strategy category deal with functional requirements, but Roohi et al. [51] and Jureta et al. [27] deal with non-functional requirements as well.

Required intervention degree: In this category, we have identified two studies [50,61] performing manual adaptation only, one study [27] performing automated adaptation only and one study [51] dealing with a hybrid approach. These results suggest that formal methods are a flexible solution regarding this aspect.

Scalability impact: No work in this category showed concerns with scalability.

Implementations: Rinderle et al. [50] present the DYCHOR framework, which exploits the semantics of the

private process changes to automatically determine the adaptations necessary for the partner processes. In these studies, message sequences are represented using annotated finite state automata (aFSA), which enables reasoning about the correctness of choreography definitions and changes. Wombacher [61] has a partial implementation of the proposed approach. Roohi et al. [51] present an implementation to check the realizability of a choreography encoded into the FSP process algebra and to automatically generate Java code for the corresponding peers for rapid prototyping purposes.

Underlying models: We have found studies using annotated finite state automata (aFSA) [50,61], nested word automata (NWA) [61], FSP process algebra [51], and Chor calculus[51].

4.5 Semantic reasoning-based approaches

Studies in this category make use of ontologies to reason about the communication among services and their replaceability. They are frequently used to foster interoperability in cross-organizational business contexts. In the Fernandez-Llatas et al. study [21], for example, the goal is to enable the communication among a set of heterogeneous sensors that are choreographed within a factory environment.

According to Fernandez-Llatas et al. [21], the use of semantic information enables the services to properly interpret the semantic meaning of the data sent by other services. However, this semantic reasoning depends on sharing ontologies across domain boundaries, which is not always possible.

Two studies belongs to this category [21,41].

Fernandez-Llatas et al. [21] proposes a semantically tagged architecture based on process choreography for distributed management of sensors. The architecture is based on the deployment of a semantic layer in choreography models to improve the configurability of the intercommunication among services.

Nabuco et al. [41] propose a self-healing ontology to describe a self-healing model, creating a shared "healing vocabulary" to project developers. This work is part of the WS-Diamond project for web services monitoring and diagnosis. The model deals with monitoring, diagnostics, healing actions, and goals. There are two levels of monitoring addressed: process and interaction. Process level deals with BPEL flow, specified using SH-BPEL (Self-Healing BPEL), and is related to diagnostic of faulty services mainly by tracking timeouts and problems during a call to a service. Interaction level deals with message exchange, by intercepting messages, extending headers with QoS parameters and processing content of SOAP body, thus dealing with diagnostic of semantic and QoS faults. The recovery process acts both upon single web services and composition of web services. The recovery is performed with actions such as retrying the execution of a process activity, redoing the execution of a process activity with different parameters, updating the value of internal variables and substituting a faulty service. The main contributions of the study are the SH-BPEL engine extension, responsible for the recovery actions, and the model, which deals with monitoring, fault diagnosis, and recovery plans.

The category aspects regarding **RQ2** are as follows:

Target: Both papers aim to support functional and non-functional adaptation.

Required intervention degree: Fernandez-Llatas et al. [21] deal with automated adaptation, whereas Nabuco et al. [41] deal with both manual and automatic intervention levels. Although some healing actions are automatically performed, the generation of adapters is a semiautomatic task.

Scalability impact: No work in this category showed concerns with scalability.

Implementations: Nabuco et al. [41] offer a plug-in for BPEL engines that provides an API to the healing system to interact with the BPEL flow. There is a page[3] describing the software produced by the WS-Diamond project, including the SH-BPEL plug-in, but no downloads are available.

Underlying models: Nabuco et al. [41] provide adaptation performed on a BPEL model.

4.6 Proxy layer approaches

The proxy layer category is related to strategies that use a software layer called proxy to intercept service messages to make the necessary adaptation. Such adaptation can be to select the final target to the message or to change the message structure to make it compliant with some service interface. We have classified two papers in this category [13,55].

Cottenier and Elrad [13] provide a Contextual Aspect Sensitive Service (CASS) platform that encapsulates coordination, activity lifecycle, and context propagation in service-oriented environments. It is built upon aspect-oriented service composition, with collaboration-based design. It operates on message level, intercepting SOAP messages at the boundaries of service components, and including mechanisms to transform, compose, synchronize, and multicast the intercepted messages.

Svirskas et al. [55] presents an approach to adapt choreographies without the need to change all the endpoints' implementations, which would be a very costly activity. Change propagations are carried by proxies, which map incoming messages to the appropriate service. The approach is very similar to pure service adaptation, but service selection is taken considering requirements from different partners. According to the authors, "these requirements are contradictory in many cases and cannot be efficiently addressed in isolation." In this work, the proxy has also the

[3] http://wsdiamond.di.unito.it/.

Table 6 Study category characteristics

Category	Targets		Intervention degree		Scalability	Implem.	Models
	Functional requirem.	Nonfunc. requirem.	Manual	Automated			
Model-based	[8,14,15,20,24,34,38,53]	[8,14,53]	[8,15,34,38,53]	[14,15,20,24]	[15]	[14,24,34]	[8,14,15,20,24,34,38]
Measurement-based	[12,18]	[12,46,63]	[18]	[12,18,46,63]	[63]	[12,46,63]	
Multi-agent	[7,40,62,64]	[62,64]		[7,40,62,64]		[7,40,62]	[7,40,62]
Formal methods	[27,50,51,61]	[27,51]	[50,51,61]	[27,51]		[50,51,61]	[50,51,61]
Semantic reasoning	[21,41]	[21,41]	[41]	[21,41]		[41]	[41]
Proxy layer	[13,55]	[55]		[13,55]		[13]	[13]

tasks of maintaining the state of the conversations between services and verifying whether a message exchange occurs in accordance with choreography rules.

The category aspects regarding **RQ2** are as follows:

Target: Cottenier and Elrad [13] deal only with functional properties, while Svirskas et al. [55] deal with both functional and non-functional properties.

Required intervention degree: Both works in this category provide automatic adaptation.

Scalability impact: Svirskas et al. [55] recommends applying an established "application-level gateway" pattern to achieve scalability, but they do not further explore how to actually use such pattern within the proposed solution.

Implementations: Cottenier and Elrad [13] provide the Context Aspect Sensitive Service platform.

Underlying models: Cottenier and Elrad [13] have no model, as their approach operates on the message level. Svirskas et al. [55] strategy is independent from model, but the authors cite WS-CDL as an option to define message exchange rules.

4.7 Relations among categories

The division into categories was performed considering the most relevant aspects of each work. However, there are some overlapping characteristics among the presented adaptation strategies of different categories. Model-based strategies are mainly supported by the idea of a business-driven approach, but sometimes, the underlying model is a formal choreography representation, such as graphs or Petri nets, which are closer to formal methods. Measurement-based strategies make a comparison between the current state and a predefined choreography model. Such model can be written in the same languages used in model-based strategies, such as WS-CDL. Both multi-agent-based and measurement-based strategies need to verify the current choreography state, although multi-agent-based strategies admit each agent having partial knowledge about the choreography state. Multi-agent-based strategies may also take advantage of WS-CDL models and Spi calculus process algebra.

In Table 6, we summarize how the selected studies of each category are distributed across the adaptation aspects' set in the research questions, regarding targets (functional and non-functional requirements), required intervention degree (manual and automated), presence of a discussion about scalability impact, and existence of implementation and underlying models.

5 Discussion

In this section, we synthesize the data about the analyzed aspects and discuss terminology. We also discuss the review process carried out and threats to the validity of this study.

5.1 Synthesis of strategies aspects

To fully answer **RQ2**, we synthesized the findings across all the studies, with respect to each strategy aspect.

Adaptation to functional requirement changes has been more broadly investigated, 92 % of the studies investigated this kind of adaptation (Fig. 1). Just two studies (8 %) dealt with non-functional requirements only, but 46 % dealt with both targets. Among the studies handling non-functional requirements, only one [62] considered security issues, and only one tackled transactional issues [14], which are important aspects for large-scale choreographies involving multiple parties.

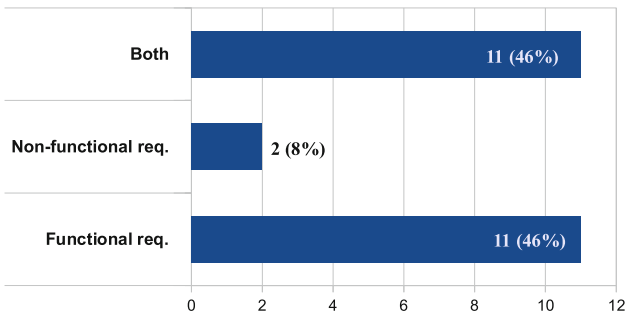

Fig. 1 Strategy targets analysis

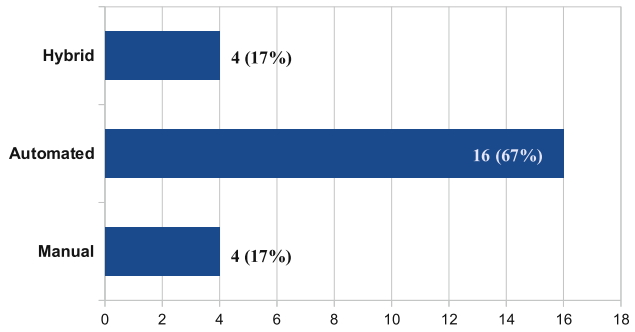

Fig. 2 Strategy human intervention degree analysis

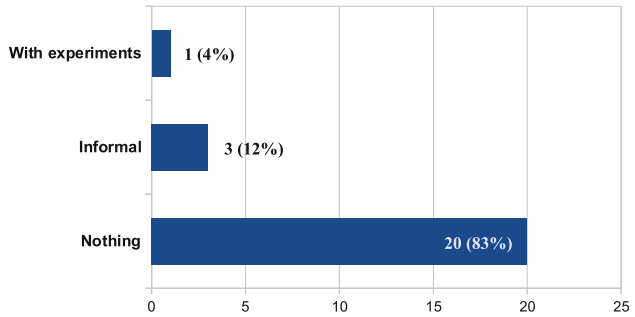

Fig. 3 Strategy impact on choreography scalability analysis

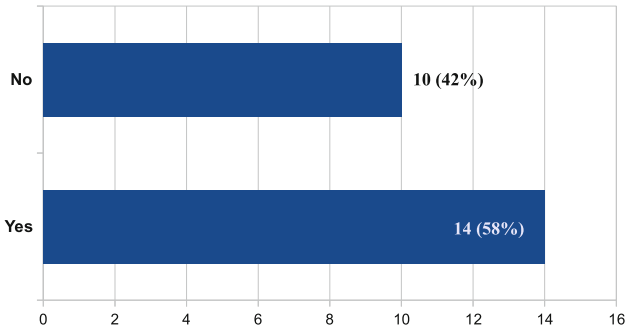

Fig. 4 Implementation analysis

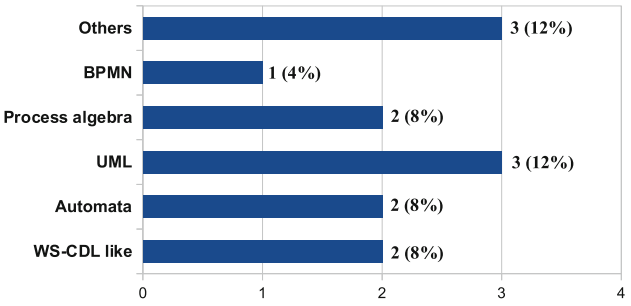

Fig. 5 Strategy underlying models analysis

Strategies with automated intervention degree are predominant (Fig. 2). Just 8 studies (33 %) considered approaches requiring human intervention to trigger or refine the adaptation. We found relationships between categories and intervention degree types: measurement-based and multi-agent-based adaptation strategies are always related to automated adaptation, although measurement-based sometimes presents hybrid mechanisms. On the other hand, strategies that present only manual adaptation are either in model-based or in formal method categories.

The scalability impact issue is missing in almost every work (Fig. 3). The single work that deals with scalability [63] uses simulation, not real implemented services. There are other three studies [15,46,55] that talk about scalability, but just in a informal way. They claim that their works provide or will provide scalability, but they do not supply evidences of such claims. The lack of scalability treatment throughout the papers is relevant, since one important claim about choreographies is that they are more scalable than orchestrations [30].

More than half of the studies (58 %) present some sort of implementation, although most of them are just prototypes (Fig. 4). Just two implementations [40,46] are available for download, and they are both released as open-source software. The Executable Process Choreography [13,14] has a release, the cassServer, but there is no license information, and the download link was broken at the time of this writing. The implementation presence is well distributed through the categories. They are present in all strategy variants: functional and non-functional requirements targets, and automated and manual intervention degrees.

The notation used is explicitly described in 10 out of the 24 studies (Fig. 5). We have grouped the found notation into six groups: BPMN, WS-CDL like, UML, process algebra, automata, and "others." The "WS-CDL like" group includes ACDL. Process algebra encompasses FSP and Spi calculus. Automata includes annotated finite state automata (aFSA) and nested word automata (NWA). Finally, the group "others" holds OperA, conversational scripts, aspect-sensitive process (ASP), and the executable choreography language (ECL).

5.2 Used terminology

The literature about distributed service composition has not yet a unified terminology, and the recent choreography technology development has posed difficulties to precise such terminology, which is a reason to recent work to claim "the need of a refined choreography notion" [52]. Table 7 describes the occurrences of the terms in the analyzed papers according to the following:

Clear—The paper clearly defines the employed term;
Inferred—The meaning of the term can be inferred in the paper;
None—The paper uses the term, but it does not explain its meaning.

Table 7 Terms related to service adaptation that were found in the selected studies

Term	Clear	Inferred	None
Adaptable/adaptive	[15,20,24,27,46,55,63]	[8,14,34,50,53,64]	[12]
Dynamic	[40,51,62]	[15]	[8,13,18,53]
Realizability	[51]		
Self-healing	[41]		
Self-configurable			[21]
Auto coordination			[21]
Runtime repair			[7]
Policies	[7]		
Agile	[20,50]		
To customize {a choreography}		[34]	[15]
Distributed adaptation reasoning			[46]
Autonomous			[18]
Awareness	[50,64]	[15]	
Context-sensitive	[13]		
On-demand deployment		[14]	

According to the table, 53 % of the listed terms were used by some paper without any definitions, whereas the same amount (53 %) was well described. The clear definitions were present at 62 % of the papers, and the usages without any definitions appeared in 42 of the papers.

We could verify that the term "adaptation," with its variants, is used with slightly different meanings. Some studies use "to adapt" as the ability to change the choreography model according to environment changes, whereas other works use "adaptation" to talk about dynamic service selection in a defined choreography model. We have noticed that there are many terms introduced with recursive statements, as *"Adaptive, mobile applications are designed to constantly adapt to the contextual conditions in an autonomous manner."* Finally, we have checked that many works use terms without proper definition.

This situation poses many difficulties on the field to compare related works. A study working on such definitions and proposing a clear terminology would be welcome.

5.3 Threats to validity

The paper selection may suffer from subjective bias. To reduce such bias, at least two reviewers participated in the selection process in each phase and a third researcher replicated some parts of it independently. Experienced researchers supervised the whole process.

We could not conduct an exhaustive search in every source. Although Goggle Scholar returned a thousand entries in response to our query string, we had examined just the 250 first entries. The last results were of low relevance for the research.

Regarding extraction bias, most part of the papers does not explicitly present the applicability context, preconditions, limitations, and drawbacks. Therefore, reviewers had to deduce and rationalize about these topics in some occasions. We evaluated aspects concerning the research method of each selected primary study. Almost every paper presented a clear description of the adaptation strategy goals. Half of the papers presented the applicability context and preconditions. Only 6 out of the 20 studies explicitly described the limitations and drawbacks of the proposed strategy. Almost half of the studies provided some kind of validation of the proposed strategy, but just four of them carried experimental validation.

Publication bias, which refers to the problem that positive results are more likely to be published than negative results, was not addressed in this systematic review. We also tried to avoid any bias related to the search string by employing as many synonyms as we could identify for the keywords. The query was also reviewed by more experienced external researchers.

Finally, we have noticed that conducting systematic reviews on emergent research topics is difficult, since a common terminology is not well established. This poses difficulties on the elaboration of the search query and the inclusion/exclusion criteria. Such difficulties led us to analyze the terms concerning adaptation that were found on the selected studies.

6 Related work

To the best of our knowledge, no systematic reviews have been produced in the specific field of service choreography

adaptation. The closest work seems to be that of Kell [28], who conducted a survey of practical software adaptation techniques. The author proposes a taxonomy for adaptation and then describes and contrasts different existing adaptation techniques.

Several adaptation studies have been conducted in the more general field of service-oriented computing. For instance, Zeng et al. [65] present AgFlow, which is a platform that applies an extensible multi-dimensional QoS evaluation model to adequately select candidate services. Furthermore, AgFlow adapts to changes that occur during the execution of a composite service in order to comply with user-defined QoS constraints. The composite service is modeled as a state chart that composes other services, and the QoS constraints are modeled using a service ontology. Van der Aalst et al. [57] demonstrate the feasibility of service behavior conformance checking by comparing message logs with service behavior specifications to detect and quantify deviations. The check is performed by mapping abstract BPEL descriptions into Petri nets that describe service behavior specifications. Although conformance checking is an important step in the life cycle of an adaptive system, it does not induce an adaptive change or determine how the adaptation occurs. Therefore, this systematic review does not cover studies dealing only with this topic.

Adaptation has also been extensively studied in the domain of workflows. According to Agostini and De Michelis [2], workflow management systems are the main technology for supporting business process. Market, social, and technological factors require frequent changes in the business process of any organization, which demands flexible workflow management systems [2]. Workflows are composed of *tasks*, which are units of work to be performed by a human or an automated agent, and *connectors* that define the order in which such tasks must be executed (*control flow*) [9]. Synchronizing concurrent executions can also be specified by means of *joins* and *forks* (also called *splits*) control flow constructs. As surveyed by Rinderle et al. [49], workflow changes can be of two types. *Instance specific changes* are applied to a single specific instance, without disturbing other instances from the same process type. *Schema changes* update all the instances of a process type. In the same work, the authors also discuss how running workflow instances can be migrated to a new workflow schema. According to Casati et al. [9], simple solutions, such as letting the processes finish according to the old model or aborting them, are often inconvenient or impossible to be applied. The work of Cicirelli et al. [11] deals with updating running instances of distributed workflow systems, where a workflow is formally modeled by a Petri net. The authors propose a decentralized Petri net execution engine that enables the deployment and enactment of a new version of an existing model without requiring to stop or remove the older instances still running. The approach uses decentralized migration procedures that migrate asynchronously portions of older instances to the new process model. In fact, the domain of distributed component-based systems also faces the problem of updating running instances. For example, Kramer and Magee [31] present a model for dynamic change management that enables leaving the updated system in a consistent state by describing just the structural changes, making the reconfiguration system independent from application. Vandewoude et al. [58] follow a similar approach, but they focus instead on how to minimize the time components get blocked in order to achieve consistent updates.

Finally, some studies have employed model-driven development to support the development of adaptive systems. According to Zhang and Cheng [66], an adaptive program needs mechanisms to ensure that the program is able to operate correctly during and after adaptations. The authors introduce a process to construct adaptation models, automatically generate adaptive programs from the models, and verify and validate the models.

7 Conclusion, research opportunities, and future work

Among other benefits, reviewing the literature helps advancing the field by seeking new lines of inquiry, identifying recommendations for further research, discovering important variables relevant to the topic, and characterizing the subject vocabulary [22]. By means of a systematic review, this study provided a summary of the state-of-the-art in service choreography adaptation. We identified 24 relevant studies and classified them into six categories according to their foci. This small number of retrieved studies corroborates our perception that service choreography dynamic adaptation is a research field not well explored.

Model-based approaches leverage techniques from model-driven development (MDD) and involve rationalizing about adaptation at a high level of abstraction. This approach is often used to empower the business analyst responsible for designing and evolving the choreography model. *Measurement-based* approaches rely on monitoring mechanisms to trigger adaptation whenever a given threshold is violated during runtime. This approach is often used to ensure a system invariant, such as average response time and availability. *Multi-agent-based* approaches use artificial intelligence techniques, such as execution plans and the BDI (beliefs, desires, and intentions) model, to define the roles and dynamics of agents. These approaches can be used to support dynamic negotiation of interaction contracts by treating choreography partners as software agents.

Formal-models-based approaches rely on process calculus and finite state machine models to describe interactions, communications, and synchronizations among services. This enables a formal reasoning about equivalences, compatibility, and replaceability of choreography services, as well as ensuring services interface conformance to a choreography model. *Semantic reasoning-based* approaches make use of ontologies to reason about the communication and replaceability of services. These approaches are frequently used to foster interoperability in cross-organizational business contexts. Finally, *proxy layer* approaches are closer to the implementation level, and suggest architectural patterns to enable service choreographies adaptation.

We analyzed adaptation strategies concerning their targets (handling functional or non-functional requirements), required intervention degree (automated or manual), scalability impact, implementations, and choreography representations (underlying models). We found out that more attention has been devoted to functional requirements and automated adaptation; only one work performs scalability evaluation; most studies present some sort of implementation and state which choreography notation is used. We have also provided some key examples of choreography dynamic adaptation usage retrieved from the selected studies. Such examples were presented within their respective adaptation strategy categories.

Research opportunities: We have noticed that some fundamental aspects are commonly ignored or poorly discussed, such as scalability, limitations, drawbacks, and evaluation. The lack of available implementations is an issue, since reproducibility is a science requirement. Security and transactions are handled by only a single study each. Furthermore, terminology is not well established. The deficiency in the treatment of these aspects raises opportunities for the improvement of existing adaptation strategies, as well as for future research in the area. Finally, we highlight that although *service selection*, *conformance checking*, and *choreography synthesis* are topics that do not directly deal with adaptation, they provide the necessary tools and techniques for a middleware to support adaptive choreographies [25]. Hence, we believe that adaptation mechanisms would benefit from deeper research into the aforementioned themes.

Future work: Future work to this systematic review encompasses (i) broadening the review scope by also considering orchestration adaptation approaches; (ii) assessing to which degree the existing implementations support real-world scenarios; (iii) and evaluating the scalability of the proposed adaptation solutions.

Acknowledgments The research leading to these results has received funding from HP Brasil under the Baile Project and from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 257178 (project CHOReOS—Large Scale Choreographies for the Future Internet). Marco Gerosa receives individual grant from CNPq.

References

1. Abadi M, Gordon AD (1997) A calculus for cryptographic protocols: the SPI calculus. In: Proceedings of the 4th ACM conference on computer and communications security. ACM, pp 36–47
2. Agostini A, De Michelis G (2000) Improving flexibility of workflow management systems. In: Business process management. Lecture Notes in Computer Science, vol 1806. Springer, pp 289–342
3. Baeten JCM (2005) A brief history of process algebra. Theor Comput Sci 335:131–146
4. Barker A, Walton CD, Robertson D (2009) Choreographing web services. IEEE Trans Serv Comput 2(2):152–166
5. Beecham S, Baddoo N, Hall T, Robinson H, Sharp H (2008) Motivation in software engineering: a systematic literature review. Inf Softw Technol 50(9–10):860–878
6. Ben Hamida A, Kon F, Ansaldi Oliva G, Dos Santos C, Lorré JP, Autili M, De Angelis G, Zarras A, Georgantas N, Issarny V, Bertolino A (2012) An integrated development and runtime environment for the future internet. In: The future internet. Lecture Notes in Computer Science, vol 7281. Springer, pp 81–92
7. Blanchet W, Elio R, Stroulia E (2005) Conversation errors in web service coordination: Run-time detection and repair. In: Proceedings of the 2005 IEEE/WIC/ACM international conference on web intelligence. IEEE, pp 442–449
8. Bræk R, Castejón H, Le H, Rossebø J (2005) Policy-based service composition and recommendation. In: Service intelligence and service science: evolutionary technologies and challenges. Addison Wesley, pp 1–20
9. Casati F, Ceri S, Pernici B, Pozzi G (1998) Workflow evolution. Data Knowl Eng 24(3):211–238
10. Cavallaro L, Di Nitto E (2008) An approach to adapt service requests to actual service interfaces. In: Proceedings of the 2008 international workshop on software engineering for adaptive and self-managing systems, SEAMS '08 ACM, pp 129–136
11. Cicirelli F, Furfaro A, Nigro L (2010) A service-based architecture for dynamically reconfigurable workflows. J Syst Softw 83(7):1148–1164
12. Colman A, Pham L, Han J, Schneider J (2006) Adaptive application-specific middleware. In: Proceedings of the 1st workshop on middleware for service oriented computing. ACM, pp 6–11
13. Cottenier T, Elrad T (2005) Dynamic and decentralized service composition. In: Proceedings web information systems and technologies. INSTICC Press, pp 56–63
14. Cottenier T, Elrad T (2005) Engineering distributed service compositions. In: Proceedings of the first international workshop on engineering service compositions, WESC'05. IBM, pp 51–58
15. Dar K, Taherkordi A, Rouvoy R, Eliassen F (2011) Adaptable service composition for very-large-scale internet of things systems. In: Proceedings of the 8th middleware doctoral symposium, MDS '11. ACM, pp 2:1–2:6
16. Di Nitto E, Ghezzi C, Metzger A, Papazoglou M, Pohl K (2008) A journey to highly dynamic, self-adaptive service-based applications. Autom Softw Eng 15(3):313–341
17. Dybå T, Dingsøyr T (2008) Empirical studies of agile software development: a systematic review. Inf Softw Technol 50(9–10): 833–859
18. Ezenwoye O, Busi S, Sadjadi SM (2010) Dynamically reconfigurable data-intensive service composition. In: Proceedins of the 6th international conference on web information systems and technologies. Springer, pp 125–130

19. Ezenwoye O, Tang B (2010) Monitoring decentralized interacting services with a global state choreography model. In: Proceedings of 8th international conference on web services IEEE, pp 671–672
20. Fabra J, Peña J, Ruiz-Cortés A, Ezpeleta J (2008) Enabling the evolution of service-oriented solutions using an UML2 profile and a reference Petrinets execution platform. In: Proceedings of 3rd international conference on internet and web applications and services. IEEE, pp 198–204
21. Fernandez-Llatas C, Mocholi JB, Moyano A, Meneu T (2010) Semantic process choreography for distributed sensor management. In: Proceedings of the international workshop on semantic sensor web. SciTePress, pp 32–37
22. Gall M, Borg W, Gall J (1996) Educational research: an introduction. Longman Publishing
23. Gu Q, Lago P (2009) Exploring service-oriented system engineering challenges: a systematic literature review. Serv Oriented Comput Appl 3(3):171–188
24. Hiel M, Aldewereld H, Dignum F (2010) Ensuring conformance in an evolving choreography. In: Proceedings of IEEE 2010 international conference on service-oriented computing and applications. IEEE, pp 1–4
25. Issarny V, Georgantas N, Hachem S, Zarras A, Vassiliadist P, Autili M, Gerosa M, Hamida A (2011) Service-oriented middleware for the future internet: state of the art and research directions. J Internet Serv Appl 2(1):1–23
26. Jorgensen M, Shepperd M (2006) A systematic review of software development cost estimation studies. IEEE Trans Softw Eng 33(1):33–53
27. Jureta I, Faulkner S, Thiran P (2007) Dynamic requirements specification for adaptable and open service-oriented systems. In: Proceedings of the 5th international conference on service-oriented computing. Springer, pp 270–282
28. Kell S (2008) A survey of practical software adaptation techniques. J Univers Comput Sci 14(13):2110–2157
29. Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Technical Report, EBSE 2007–001, University Joint Report. Keele University and Durham
30. Kokash N, D'Andrea V (2005) Service oriented computing and coordination models. In: Proceedings of challenges in collaborative engineering workshop. Citeseer, pp 95–103
31. Kramer J, Magee J (1990) The evolving philosophers problem: dynamic change management. IEEE Trans Softw Eng 16(11):1293–1306
32. Lesser V (2003) Multi-agent Systems. In: Encyclopedia of computer science, 4th edn. Wiley, pp 1194–1196
33. Liu D, Law KH, Wiederhold G (2002) Analysis of integration models for service composition. In: Proceedings of the 3rd international workshop on software and performance, WOSP '02. ACM, pp 158–165
34. Mahfouz A, Barroca L, Laney R, Nuseibeh B (2009) Requirements-driven collaborative choreography customization. In: Proceedings of the 7th international joint conference on service-oriented computing Springer, pp 144–158
35. McKinley P, Sadjadi S, Kasten E, Cheng B (2004) Composing adaptive software. IEEE Comput 37(7):56–64
36. Mellor SJ, Clark AN, Futagami T (2003) Guest editors' introduction: model-driven development. IEEE Softw 20:14–18
37. Mian P, Conte T, Natali A, Biolchini J, Travassos G (2007) A systematic review process for software engineering. Empir Softw Eng 32(3):1–6
38. Moo-Mena F, Drira K (2007) Modeling architectural level repair in web services. In: Proceedings of the 3rd international conference on web information systems and technologies. Springer, pp 240–245
39. Morreale V, Bonura S, Francaviglia G, Cossentino M, Gaglio S (2005) PRACTIONIST: a new framework for BDI agents. In: Proceedings of the 3rd European workshop on multi-agent systems, pp 236–247
40. Morreale V, Puccio M, Cammarata G, Francaviglia G (2007) Dynamic conversations between agents with the PRACTIONIST framework. In: Proceedings of 5th IEEE international conference on industrial informatics. IEEE, pp 1065–1070
41. Nabuco O, Halima R, Drira K, Fugini M, Modafferi S, Mussi E (2008) Model-based QoS-enabled self-healing web services. In: Proceedings of the 19th international conference on database and expert systems application. IEEE, pp 711–715
42. Nanda MG, Chandra S, Sarkar V (2004) Decentralizing execution of composite web services. In: Proceedings of the 19th annual ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications, OOPSLA '04 ACM, pp 170–187
43. OASIS: Web services business process execution language (WS-BPEL), version 2.0 (2007). http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
44. OMG: business process model and notation (BPMN), version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0
45. Papazoglou MP, Traverso P, Dustdar S, Leymann F (2007) Service-oriented computing: state of the art and research challenges. IEEE Comput 40(11):38–45
46. Paspallis N, Papadopoulos G (2006) Distributed adaptation reasoning for a mobility and adaptation enabling middleware. In: On the move to meaningful internet systems 2006: OTM 2006 workshops, Springer, pp 17–18
47. Pedraza G, Estublier J (2009) Distributed orchestration versus choreography: the FOCAS approach. In: Proceedings of the 2009 international conference on software and systems process. Springer, pp 75–86
48. Poulin M (2011) Collaboration patterns in the SOA ecosystem. In: Proceedings of the 3rd workshop on behavioural modelling. ACM, pp 12–16
49. Rinderle S, Reichert M, Dadam P (2004) Correctness criteria for dynamic changes in workflow systems—a survey. Data Knowl Eng 50(1):9–34
50. Rinderle S, Wombacher A, Reichert M (2006) Evolution of process choreographies in DYCHOR. In: On the move to meaningful internet systems 2006: CoopIS, DOA, GADA, and ODBASE. Springer, pp 273–290
51. Roohi N, Salaün G, France V (2011) Realizability and dynamic reconfiguration of Chor specifications. Inf int J Comput Inf 35(1):39–49
52. Schönberger A (2011) Do we need a refined choreography notion? In: Proceedings of the 3rd central-European workshop on services and their composition, ZEUS, CEUR workshop proceedings, vol 705, pp 16–23. CEUR-WS.org
53. Stegaru G, Stanescu AM, Sacala I, Moisescu M (2012) Dynamic interoperability model for web service choreographies. In: Enterprise interoperability V. Proceedings of the I-ESA conferences, vol 5. Springer, pp 81–91
54. Steinmacher I, Chaves AP, Gerosa MA (2010) Awareness support in global software development: a systematic review based on the 3C collaboration model. In: Proceedings of the 16th international conference on Collaboration and technology, Springer, pp 185–201
55. Svirskas A, Roberts B, Ignatiadis I (2008) Adaptive service choreography support in virtual enterprises. In: Agent and web service technologies in virtual enterprises. IGI Global, pp 66–74
56. Sycara KP (1998) Multiagent systems. AI Mag 19(2)
57. Van der Aalst WMP, Dumas M, Ouyang C, Rozinat A, Verbeek E (2008) Conformance checking of service behavior. ACM Trans Internet Technol 8(3):13:1–13:30

58. Vandewoude Y, Ebraert P, Berbers Y, D'Hondt T (2007) Tranquility: a low disruptive alternative to quiescence for ensuring safe dynamic updates. IEEE Trans Softw Eng 33(12):856–868
59. W3C: Web service choreography interface (WSCI) (2002) version 1.0. http://www.w3.org/TR/2002/NOTE-wsci-20020808
60. W3C: Web services choreography description language (WS-CDL) (2005) Version 1.0. http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109
61. Wombacher A (2009) Alignment of choreography changes in BPEL processes. In: Proceedings of IEEE 2009 international conference on services computing. IEEE, pp 1–8
62. Xu D, Qi Y, Di Hou Y, Liu L (2007) A formal model for dynamic web services composition MAS-Based and simple security analysis using SPI calculus. In: Proceedings of the 3rd international conference on next generation web services practices. IEEE, pp 69–72
63. Yang L, Dai Y, Zhang B (2009) Performance prediction based EX-QoS driven approach for adaptive service composition. J Inf Sci Eng 25(2):345–362
64. Yau S, Huang D, Gong H, Davulcu H (2005) Situation-awareness for adaptive coordination in service-based systems. In: Proceedings of the 29th annual international computer software and applications conference. IEEE, pp 107–112
65. Zeng L, Benatallah B, Ngu A, Dumas M, Kalagnanam J, Chang H (2004) QoS-aware middleware for web services composition. IEEE Trans Softw Eng 30(5):311–327
66. Zhang J, Cheng BHC (2006) Model-based development of dynamically adaptive software. In: Proceedings of the 28th international conference on software engineering, ICSE '06, ACM, pp 371–380