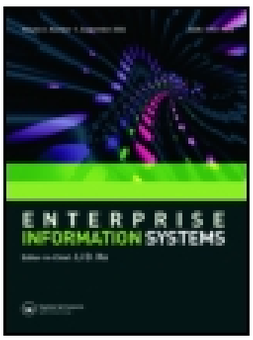

Enterprise Information Systems

ISSN: 1751-7575 (Print) 1751-7583 (Online) Journal homepage: https://www.tandfonline.com/loi/teis20

A methodology to rank enterprise application integration platforms from a performance perspective: an analytic hierarchy process-based approach

Daniela L. Freire, Rafael Z. Frantz, Fabricia Roos-Frantz & Sandro Sawicki

To cite this article: Daniela L. Freire, Rafael Z. Frantz, Fabricia Roos-Frantz & Sandro Sawicki (2019): A methodology to rank enterprise application integration platforms from a performance perspective: an analytic hierarchy process-based approach, Enterprise Information Systems, DOI: 10.1080/17517575.2019.1633692

To link to this article: https://doi.org/10.1080/17517575.2019.1633692

Published online: 30 Jun 2019.

ENTERPRISE INFORMATION SYSTEMS
https://doi.org/10.1080/17517575.2019.1633692

Taylor & Francis
Taylor & Francis Group

ARTICLE TEMPLATE

A methodology to rank enterprise application integration platforms from a performance perspective: an analytic hierarchy process-based approach

Daniela L. Freire, Rafael Z. Frantz, Fabricia Roos-Frantz and Sandro Sawicki

Department of Exact Sciences and Engineering, Unijuí University, Ijuí, Brazil

ABSTRACT

Companies' software ecosystems comprise applications that support their business processes. Frequently, these applications have been developed with different technologies and with no concern about integration. Integration platforms are tools that facilitate the development and execution of integration solutions. In this article, we propose a methodology to support software engineers in the decision-making process for an integration platform when performance is a central requirement. The proposed methodology adds objective criteria for performance assessment purposes and has been used to rank the five most popular integration platforms in order to prove its feasibility.

ARTICLE HISTORY
Received 12 November 2018
Accepted 16 June 2019

KEYWORDS
Application integration; integration platform; integration patterns; analytic hierarchy process; multiple criteria decision-making; methodology

1. Introduction

Software ecosystems (Manikas 2016) are formed over time by the incorporation of their own applications or applications purchased from other companies. Recently, cloud computing services have also been included (Varghese and Buyya 2018), making these ecosystems more heterogeneous. Such applications are designed to operate locally and on embedded systems or clusters of cloud computing, running on a number of processors and dealing with a large volume, variety and velocity of data that characterises the big data scenario (Ritter, May, and Rinderle-Ma 2017). Nevertheless, such applications are not designed to work together nor collaborate on exchanging data and sharing functionality to support business processes.

Enterprise application integration is a research field that provides methodologies, techniques and tools for the modelling and implementation of integration solutions (Freire, Frantz, and Roos-Frantz, 2019). Such solutions promote the applications' orchestration in order to keep data synchronised or to develop new functionality on top of what currently exists, with minimal impact (Frantz, Corchuelo, and Roos-Frantz 2016). Integration platforms are tools that allow for the design and execution of these solutions. Such tools usually provide a domain-specific language, a development toolkit, a run-time system and monitoring tools. The domain-specific language enables the description of conceptual models for integration solutions. The development toolkit is a set of software tools for the implementation of solutions, e.g., transforming

CONTACT Daniela L. Freire. dsellaro@unijui.edu.br Department of Exact Sciences and Engineering, Unijuí University, Brazil

a conceptual model into executable code.The monitoring tools detect errors that may occur during the execution of integration solutions. The run-time system is responsible for running integration solutions, which makes its performance a critical issue (Khoumbati, Themistocleos, and Irani 2006; Botta et al. 2016; Ebert, Weber, and Koruna 2017). It is commonplace for open-source integration platforms to adopt the conceptual integration patterns, as documented by Hohpe and Woolf (2004), and the architectural style of pipes and filters (Alexander, Ishikawa, and Silvertein 1977). The integration patterns are guidelines for implementing tasks that help to solve recurrent problems associated with integration application. In the pipe-and-filter architectural style, filters implement tasks that provide an integration pattern, with the pipes implementing communication channels through which the data flow wrapped in messages. Example of open-source integration platforms are as follows: Spring Integration (Fisher et al. 2012), Camel (Ibsen and Anstey 2010), Fuse (Russell and Cohn 2012), Petals (Surhone, Timpledon, and Marseken 2010) and Guaraná (Frantz, Corchuelo, and Molina-Jiménez 2012).

Enterprises have sought to optimise and integrate their business processes by increasing the usage of software applications to support them. One alternative for small and medium-sized enterprises has been to hire the integration platform as a service (iPaaS), because such a service decreases concerns about maintenance costs and operating integration platforms on their premises (Brahmi and Gharbi 2014). However, these platforms need to meet modern demands, while their run-time systems must be increasingly robust, resilient and flexible, thus allowing integrated applications to provide response times in milliseconds and uptime availability (Harman et al. 2013; Linthicum 2017; Ritter, May, and Rinderle-Ma 2017). Due to the existence of a large number of alternatives for integration platforms in the market, choosing the best one is not an easy task. We consider that the run-time system of an integration platform is more effective if it consumes fewer computational resources and is able to process more messages per unit of time. In our review of the literature, we have identified some proposals that can help software engineers to compare integration platforms in terms of performance. Most of them focus on certain market aspects such as company maturity, customer relationships and innovation, alongside technology aspects such as the availability of connectors, security, monitoring and governance of integration solutions.

In this article, we introduce a multiple criteria decision-making methodology, based on the analytic hierarchy process (AHP) method (Saaty 1990), which has been widely used in decision-making processes when choosing applications (Vaidya and Kumar 2006). It makes use of mathematical and psychological fundamentals: the former organise and analyse complex decisions, the latter help to perform human judgements (Saaty 2008). There are three main steps in the proposed methodology: (a) evaluating comparison properties, (b) ranking construction, (c) data visualisation and interpretation. Step (a) for evaluating comparison properties defines the performance properties and their respective values, which comprise three dimensions: message processing, hotspot detection and fairness execution-related properties. Message processing aims to improve the efficiency of the run-time system in order to process a message, while, in hotspot detection, the focus is on the detection of tasks that may represent a bottleneck within an integration solution. In turn, fairness execution deals with the minimisation of the average time that a message takes to be processed by a fair assignment of computational resources. Step (b), which involves ranking construction, produces the

ranking for the integration platforms. It is divided into four activities: dimension relevance, platform competence, score computation and pairwise comparison. Step (c), which comprises data visualisation and interpretation, presents and discusses the results. The platforms are evaluated by subjective and objective criteria. In the former case, this is carried out according to experts' judgements while the latter is carried out by the assignment of scores for performance properties. The contribution of the values assigned to the objective criteria is greater than the values attributed to the subjective criteria in the composition of the platform score.

This methodology is the result of our experience, over a period of several years, of developing integration projects in real-world software ecosystems where performance is a key feature. Our goal is to help software engineers in their choice of an integration platform based on performance. This methodology is applicable to the evaluation and comparison of any other set of integration platforms (see Freire, Frantz, and Roos-Frantz (2019) for an experience report on its application). The methodology has practical implications for software engineers by ranking platforms from the perspective of the performance of the run-time system in the execution of an integration solution. Software engineers can take advantage of the respective set of properties in order to evaluate the performance of integration platforms and base their choice on objective criteria, in addition to their own knowledge as experts. The proposed methodology is simple because it has only a few steps, complemented by detailed descriptions and templates for data aggregation and presentation. It is a flexible methodology because it allows for the prioritisation of dimensions based on the company's interests. It can also be applied to other platforms and supports the use of any support tools. Additionally, its basis on the AHP method ensures consistent results. This article is organised as follows. Section 2 discusses related works on methodologies to evaluate and compare integration platforms. Section 3 introduces our methodology, as well as describes its steps and related activities. Section 4 demonstrates this methodology in action by evaluating, comparing and ranking five well-known open-source integration platforms. Lastly, Section 5 presents our conclusions regarding the methodology and its application.

2. Related work

In this section, we discuss related works that present methodology or approach subjects that intersect the research about performance analysis, comparison or ranking of integration platforms.

Kawaguchi and Yamada (2005) proposed a simulation method to evaluate the performance of application integration based on a timeout control scheme. They approached factors that can affect the integration, such as network, application traffic, and workflow layer models. The method measured the total service completion time and the individual application response time as performance measures. In their work, they used a simulation tool, called OPNET, and their contribution was the improvement of the tool, whereas this article provides a methodology that approaches the performance of integration platforms and our contribution is to provide well-defined steps and templates to support the decision-making in the choice of the best platform by software engineers.

Corchuelo, Frantz, and González (2008) analysed integration platforms regarding platform independence, usability, ease of programming, and maintainability. They divided the properties into three groups: scope of the tool, modelling capabilities, and technical features. The scope of the tool deals with properties that the authors consider as essential. The modelling capabilities deals with important properties but not essential; according to the authors, the absence of such properties makes integration modelling more complex and less intuitive. The technical features address properties that affect the ease of programming, performance, or management of integration solutions. In their work, they compared five integration platforms: Camel, Mule, ServiceMix, Spring Integration, and BizTalk. Their work differs from ours, mainly because their proposal focuses on general properties, whereas our proposal provides a decision-making method to compare integration platforms regarding performance.

Santos, Sarriegi, and Serrano (2008) proposed a methodology of integrated information systems design to help enterprises to align management applications with their business processes and to define the integration of the information systems of enterprises. The methodology has two main steps. First step defines a functional reference model of the company and the second applies a generic functional model, which covers only the main processes. Their work presents a methodology to help enterprises to represent their core processes and applications involved in these processes, whereas this article provides a methodology to help them to evaluate tools to integrate their applications.

Aldin and de Cesare (2011) proposed a methodology to discover process patterns from diverse enterprises assets and to model such patterns in order to reuse them in the design or redesign of organisational processes. The literature surveyed and the themes that can be drawn from their work aimed to define a methodology for systematically discovering and using generalised patterns of business processes; whereas, in this article, we present a decision-making methodology for integration platforms regarding performance aiming improvements for the enterprises' business process.

Tan et al. (2011) analysed the design of integration solutions that make use of the Environment-based Design methodology, in which a design problem is implied in a product system. Their methodology is composed of three parts: the environment of the product project, the requirements on product structure, and the requirements on the performance of the product project. Their methodology aims to generate and to refine the design specifications and design solutions, following three steps that are carried out progressively and simultaneously: environment analysis, conflict identification, and solution generation. The «environment analysis» identifies the key environment components and the relationships amongst them. The «conflict identification» identifies conflicts within the environment components relationships. The «solution generation» solves environment conflicts, proposing a design solution. Their work used a methodology to eliminate conflicts in designs of integration solutions, whereas we propose a methodology to evaluate the performance of execution of integration solutions.

Traore et al. (2012) introduced a model-driven process that assists in performance analysis throughout the software development life-cycle, allowing designing models for the performance analysis of distributed software systems, based on the Unified Modelling Language profile for schedulability, performance and time. Their methodology provided an outline of system performance models, metrics, and a case study of a business system. Their

work presented a methodology for the evaluation of distributed systems, whereas we propose a methodology to the comparison of integration platforms.

More and Bartere (2013) analysed features that impact the productivity, scalability, elasticity, cycle time and incorporated in quality of IT teams, reporting their experiences about the use of the Amazon SQS and Boomi integration platforms. These authors pointed out the pros and cons of each platforms, suggesting the Boomi platform for projects of lower costs and when there is a lack of seasoned software engineers. They analysed and compared only two integration platforms and did not describe the steps followed, so that the procedure can not be repeated, while we describe in detail the steps of the methodology to evaluate integration platforms.

Roos-Frantz et al. (2015) proposed simulation like an evaluation method of the quality of conceptual models of integration solutions. They used Petri nets to simulate integration solutions conceptual models designed with Guaraná platform (Frantz, Corchuelo, and Molina-Jiménez 2012). The authors claim that this method allows cost reduction, risk, and time because this approach precedes the construction of the real solution, avoiding future fails. Their work presents a method to evaluate the conceptual models of integration solutions, whereas this article provides a methodology to evaluate live integration platforms with the focus in performance of the execution of integration solutions. Reports published by consulting and marketing trends companies often review and compare integration platforms.

Ebert and Weber (2016) analysed and evaluated integration platforms by a taxonomy, and applied it to the following platforms: Boomi, Informatica, Mule, and SAP. They used two groups of criteria: functional and non-functional. The functional criteria were processed execution, number of operators, connectivity, administration, and development. The non-functional criteria studied were: price, service contract, trustworthiness, technology, safety, and service management. Their work had a general purpose approach, whereas this article has specific purposes for a decision-making method to rank integration platforms.

The reports by Gartner (Guttridge et al. 2017) and Ovum (Sharma 2017) focus on properties related with market aspects, such as provider company maturity, customer relationship, or innovation, and others related with technology, such as the availability of connectors, security, monitoring, and governance of integration solutions. These reports provide a view of the integration platforms in terms of the market, whereas this article focuses on properties that have an impact on the performance of the run-time system and indicate a decision-making method to choose an integration platform, focusing on the performance.

Pfaff and Krcmar (2018) proposed a web-based system architecture for data integration that allows numerous external data sources to be linked with the domain ontology. Their research suggested to structure, standardise and normalise IT service catalogues. Their work proposed an architecture to integrate data, whereas our proposal is to evaluate integration platforms. The related works are summarised in Table 1, specifying the kind of methodology used, the kind of approach, and the analysed properties.

3. The ranking methodology

This section presents our methodology for decision-making by comparing and ranking enterprise application integration platforms and focusing on the performance of their run-

Table 1. Comparison between approaches in the related works.

Proposals	Methodology	Approach	Analysed Properties
Kawaguchi and Yamada (2005)	simulation	application integration	network, application traffic, and workflow layer models
Corchuelo, Frantz, and González (2008)	n/a	integration platform	independence, usability, easiness of programming, and maintainability
Santos, Sarriegi, and Serrano (2008)	model	organisational process	management applications
Aldin and de Cesare (2011)	model	organisational process	process patterns
Tan et al. (2011)	model	integration solutions	environment of the product project
Traore et al. (2012)	model	development process	life-cycle steps
More and Bartere (2013)	n/a	IT team	productivity, scalability, elasticity, cycle time and baked in quality
Roos-Frantz et al. (2015)	simulation	integration solution	quality of conceptual models
Ebert and Weber (2016)	taxonomy	integration platform†	processed execution, number of operators, connectivity, administration, development, price, service contract, trustworthiness, technology, safety, and service management
Guttridge et al. (2017)	n/a	integration platform†	provider company maturity, customer relationship, or innovation
Sharma (2017)	n/a	integration platform†	availability of connectors, security, monitoring, and governance of integration solution
Pfaff and Krcmar (2018)	architecture	data integration	structure, standardise and generalise of IT service
Our proposal	model	run-time system	thread pool creation and configuration, message storage, detection stage, abstraction level, pattern identification, execution model, scheduling policy, and throttling controller

† Performance was approached in high-level of abstraction.

time systems. The following steps constitute this methodology: evaluating comparison properties, ranking construction, and data visualisation and interpretation, cf. Figure 1.

The «evaluating comparison properties» step deals with the study of the run-time system and results in a set of values assigned to the properties, which can have an impact on performance. Such properties are grouped according to three dimensions, which will be described later. The «ranking construction» step deals with the production of the ranking of integration platforms and is divided into the following activities: (a) dimension relevance, (b) platform competence, (c) score computation and (d) pairwise comparison. The (a) dimension relevance activity consists of determining the relevance level of each dimension in the context of integration platform usage. The (b) platform competence activity consists of determining the capacity of the integration platform to find the properties of each dimension, using objective and subjective criteria. The (c) score computation activity calculates the scores for integration platforms based on the results of previous activities. The (d) pairwise comparison activity is used to evaluate to what extent one integration platform is better than another regarding each dimension. Finally, the «data visualisation and verification» step focuses on the analysis and presentation of the results. Our methodology uses the *TransparentChoice*[1] software as a support tool whose graphic interface helps in the prioritisation of dimensions and in the process by which software engineers make judgements and carry out consistency tests for evaluation. We chose this tool because it offers a free version and because there was knowledge about its use amongst the researchers. However, there are several AHP tools in the market, such as *Expert Choice*,[2] *SuperDecisions*,[3] *AHP-OS*,[4] *Decisor*[5] czekster et al. 2019, and *PriEsT*.[6] Further, this methodology allows software engineers to adopt the tool that suits them best, because there is no dependency on any of them.

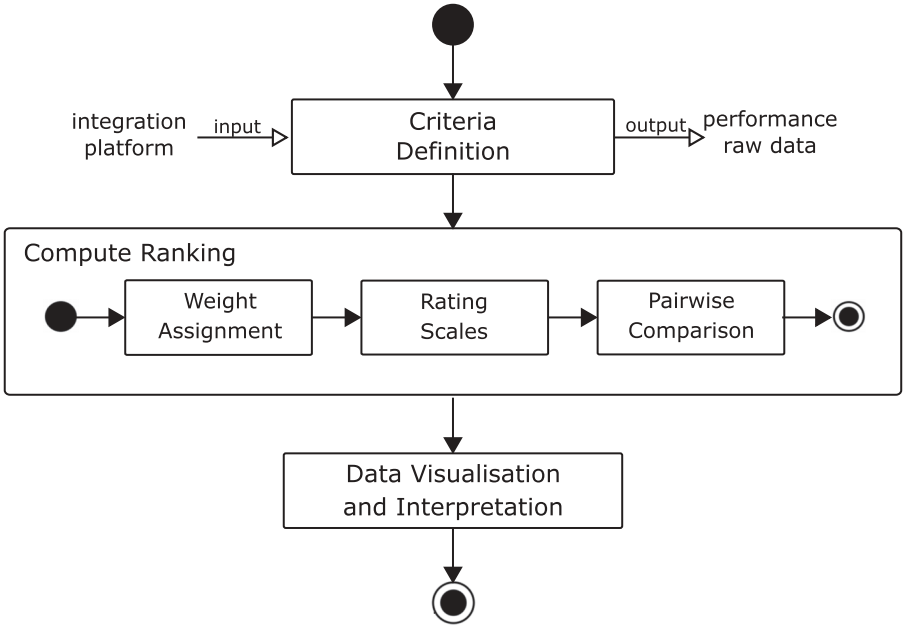

Figure 1. Overview of the proposed methodology.

3.1. Evaluating comparison properties

In this step, the integration platforms are carefully studied, so that their run-time systems are evaluated regarding properties that can affect performance. These properties represent the evaluation criteria in our methodology and are organised in three dimensions: (i) message processing, (ii) hotspot detection and (iii) fairness execution. The (i) message processing dimension pursues efficient message processing within an integration solution. The (ii) hotspot detection dimension aims to discover the hotspot within an integration solution, in which the message processing time breaches the software quality requirements of the company in question. The (iii) fairness execution dimension seeks to distribute the computational resources in a balanced way during message processing dimension within an integration solution. In this article, we refer to threads as computational resources for executing integration solution task in message processing. A thread is the basic unit of processing, i.e., the smallest sequence of programmed instructions that can be managed by the run-time system. We introduced the dimensions and their corresponding properties and provided candidate values to each property, so that software engineers can assign the appropriate value to the property of the integration platform analysed.

Software engineers summarise the evaluation results concerning integration platforms in the template shown in Table 2. In this table, the columns represent the integration platforms, the lines represent the properties of each of the dimensions and the cells represent the qualitative values of the properties.

3.1.1. Message processing

This dimension addresses improvements to the run-time system's efficiency when processing a message. It can also be seen as an increase in the number of processed messages per unit of time. It comprises properties that reduce the real time demanded by the integration solutions to process a message completely. These properties are:

3.1.1.1. Thread pool creation. This property indicates how the run-time system makes the thread pools available, in order to execute the integration solutions, by allowing for the creation of one or more thread pools (Ibsen and Anstey 2010). It may assume the following values, in ascending order of preference: global or local. The global value indicates that a single thread pool executes every task in an integration solution. The

Table 2. Template for summarising the values of comparison properties.

		Integration Platforms		
Dimension	Property	*Platform 1*	...	*Platform n*
Message Processing	Thread pool creation	*	...	*
	Message store	*	...	*
	Thread pool configuration	*	...	*
Hotspot Detection	Abstraction level	*	...	*
	Detection stage	*	...	*
	Pattern identification	*	...	*
Fairness Execution	Execution model	*	...	*
	Scheduling policy	*	...	*
	Throttling controller	*	...	*

local value indicates that the run-time system is able to configure the local thread pools in order to execute a task or a task group for an integration solution.

3.1.1.2. Message storage. This property indicates how the run-time system deals with the storage of messages during the execution of an integration solution. In-memory storage of messages is faster (Balko and Barros 2015), but can be more expensive. Messages that contain big amounts of data have an impact on the amount of memory required for their processing inside the integration solutions. In such cases, rather than only in-memory storage of messages, the run-time system can store them on disk. It may assume the following values, in ascending order of preference: in-memory or hybrid. The in-memory value indicates that the run-time system only performs the in-memory storage of messages. The hybrid value indicates that the run-time system adopts different strategies for storing messages, in addition to in-memory storage.

3.1.1.3. Thread pool configuration. This property indicates how the run-time system tunes the thread pool size to deal with different message workloads. The run-time system allows for the configuration and management of thread pools for an integration solution. It may assume the following values, in ascending order of preference: static or dynamic. The static value indicates that the thread pool is configured with a fixed number of threads, as defined at the design stage by the software engineer. The dynamic value indicates that, during run-time, the run-time system dynamically tunes the number of threads in the pool, whether increasing or decreasing, according to the demand of the workload within a range of values established at the design stage (Gleyzer and Howes 2017).

3.1.2. Hotspot detection

This dimension addresses the hotspot detection within the integration solutions according to the run-time system. A hotspot is part of an integration solution where there is a bottleneck; therefore, messages are not processed in an acceptable time. They are characterised by the task's work overload and indicate whether threads are lacking for that point in the integration solutions. This can lead to an increase in the actual time required to process a message. Hotspot detection can help the run-time system allocate threads more efficiently to tasks in the integration solutions. The following properties can contribute to hotspot detection:

3.1.2.1. Detection stage. This indicates the stage at which it is possible to determine the existence of hotspots in the integration solutions (Ahmad 2016). This property may assume the following values, in ascending order of preference: design time, run-time. If the detection stage assumes the design time value, this means that software engineers must use their domain knowledge and modelling experience to forecast hotspots during the design stage, whereas run-time means that the run-time system is endowed with intelligence needed to detect hotspots during the execution of the integration solutions.

3.1.2.2. Abstraction level. This property indicates the level of decomposition at which the integration solutions can be broken in order to detect hotspots. Hotspot detection can be carried out by dividing the integration solution into pieces, which are either chained tasks groups or single tasks. This property may assume the following values, in

ascending order of preference: per group, per task. The per group value means that the piece of the integration solution being analysed equals a group of tasks. The per task value means that the piece of the integration solution being analysed equals a single task. This last value allows for a finer-grained control (Sudarsanam, Srinivasan, and Panchanathan 2004).

3.1.2.3. Pattern identification. This property indicates whether the run-time system can detect patterns (Ritter, May, and Rinderle-Ma 2017) that can lead to hotspots. Such patterns may determine: (i) known behaviours that occur during execution, increasing the average waiting time of the tasks that are ready to be executed; (ii) a particular combination of tasks identified in the design of the integration solutions. This property may assume the following values, in ascending order of preference: no or yes. The yes value indicates that the run-time system can detect patterns that might cause hotspots when the integration solution is divided into pieces; in cases where it does not have this ability, the value is no.

3.1.3. Fairness execution

This dimension addresses the balanced assignment of threads to tasks, in order to minimise the average time that a message takes to be processed in an integration solution. The following properties contribute to the fair execution of tasks:

3.1.3.1. Execution model. This property refers to the execution model implemented by the run-time system. It deals with the level of execution of an integration solution. It is possible to classify these models into process-based and task-based models. In the former case, the run-time system controls process instances as a whole, i.e., it cannot interact with internal tasks. In the latter case, the run-time system can control both process instances and their internal tasks. The literature reports that the task-based model offers better performance with a steady stream of data input and lower performance when the input rate increases (Frantz, Corchuelo, and Arjona 2011). This model is also more complex to provide transaction and fault-tolerance support (Frantz, Corchuelo, and Molina-Jiménez 2012). This property may assume the following values, in ascending order of preference: process-based, task-based or hybrid. The process-based value indicates that the run-time system adopts a process-based model. The task-based value indicates that the run-time system adopts a task-based model. The hybrid value indicates that the run-time system will adopt the model which best fits the execution profile in terms of predefined parameters, such as the message input rate, the number of processors or the average message size.

3.1.3.2. Scheduling policy. This property refers to the policy followed by the run-time system in order to schedule the task execution in an integration solution using computational resources. Usually, the tasks stay in a queue until there are available threads to execute them. In cloud environments, the scheduling of an integration solution becomes challenging, because its performance must result in scheduling overhead reductions, minimised costs and maximised resource utilisation, while still meeting the specified deadlines (Anwar and Deng 2018). However, cloud environments can lead to computing overheads that negatively affect the overall performance and the execution costs (Chen and Deelman

2011). This property can assume following values, in ascending order of preference: fifo, priority or mapping. The fifo value means that the run-time system follows a first-in-first-out policy, in which the oldest (first-in) task in the queue is firstly executed. The priority value means that the run-time system allows tasks to have an associated priority, so that a task with a high priority is firstly executed. The mapping value means that the run-time system invokes a mapping procedure that involves a mathematical model or optimisation method, which helps in finding an optimal scheduling policy for task execution, based on a previous evaluation of the integration solution.

3.1.3.3. Throttling controller. This property indicates whether the run-time system can control the rate of incoming messages in an integration solution, so that when this rate exceeds a previously determined limit, the run-time system can adopt suitable policies to preserve the execution of the integration solution. Such intervention policies may involve: (i) refusing new messages; (ii) buffering at input the incoming messages or maintaining them in a repository. This property may assume the following values, in ascending order of preference: no or yes. The yes value indicates that the run-time system can control the incoming message rate, i.e., it has a throttling controller; in cases where it does not have a throttling controller (Hohpe and Woolf 2004), the value will be no.

3.2. Ranking construction

This step aims to produce the results that lead to the ranking of the integration platforms. It consists of determining the relevance level of each dimension by rating the integration platforms' capacity in relation to their properties, merging the criteria to calculate the scores for the integration platforms, and performing a pairwise comparison of the integration platforms. This step is composed of the following activities: relevance assignment, competence rating, score computation and pairwise comparison.

Our methodology structures the problem at successive levels, which enable the formulation of a complex decision in a hierarchical structure in the form of an inverted tree, cf. Figure 2. The highest level is the main goal, i.e., choosing the integration platform with the best performance. At the second level are the criteria for decision-making, i.e., the dimensions. At the bottom level are the alternatives, i.e., the integration platforms.

3.2.1. Dimension relevance

This activity determines the relevance level of each dimension. The hierarchical structuring of the problem facilitates the judgement of a dimension regarding its priority in relation to the other dimensions. This prioritisation is performed by collecting the opinions of software engineers regarding the relevance of each dimension in the performance of the integration platforms. An opinion is represented by the assignment of a weight (ω) to each dimension, which determines the relevance level of the dimension and provides a consistency measure and quality indicator of the chosen integration platform. Table 3 sets out the weights that can be assigned to a dimension.

When properties of a dimension are fundamental to achieving good performance, they must be highly considered in the choice of integration platform. Thus, the relevance of this dimension is set at the *essential* level and assumes a weight equal to 3. If the dimension improves the performance, but it is not essential, then the relevance level

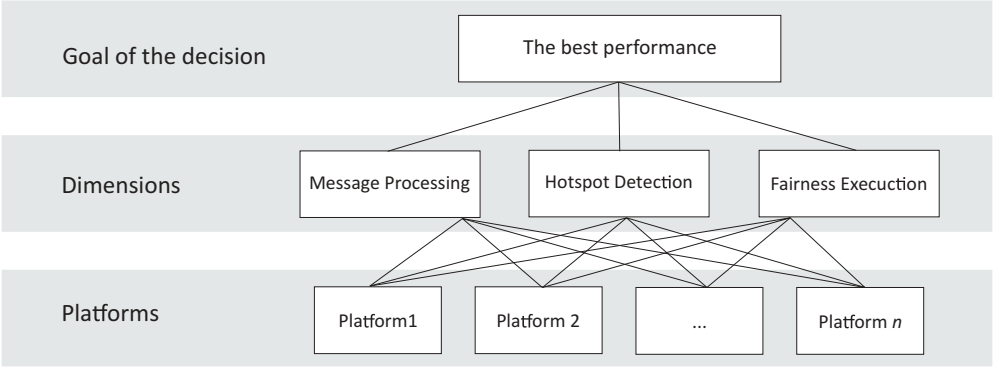

Figure 2. Hierarchical structure for the decision problem.

Table 3. Relevance levels for dimensions.

Weight (ω)	Relevance	Description
3	Essential	The dimension is essential to meet the goal.
2	Important	The dimension is important to meet the goal.
1	Desirable	The dimension contributes to the goal as an additional element.

of this dimension will be set to *important*, which assumes a weight equal to 2. If the dimension adds positive features, but its absence does not significantly affect the performance, then the relevance level of this dimension will be set to *desirable*, which assumes a weight equal to 1. Table 4 presents a template that can be used to organise the assigned relevance levels to each dimension.

3.2.2. Platform competence

This activity quantifies the integration platforms' capacity based on their properties and involves subjective or objective evaluations. At the end of this activity, each integration platform receives numeric values that correspond to the score for the competence level in each dimension, in order to realise good performance in the integration solution's execution.

Subjective evaluation follows a scale proposed by Saaty Saaty (1990), where values relating to the competence of the integration platforms vary from a minimum value of 0 to a maximum value of 9. This scale is widely used in decision-making methods and based on psychological observations on passing judgements (Franek and Kresta 2014). A minimal value equal to 0 indicates that the platform does not meet any of the properties of the evaluated dimension. A maximal value equal to 9 indicates that the platform has found all properties of the evaluated dimension. Values from 1 to 8 indicate that the platform has partially found properties of the evaluated dimension.

Table 4. Table template to summarise dimension relevance.

Dimension	Essential	Important	Desirable
Message Processing	*	*	*
Hotspot Detection	*	*	*
Fairness Execution	*	*	*

The template shown in Table 5 can be used by software engineers to register the values of competences for the integration platforms, as obtained by the subjective evaluation. This table must be populated after software engineers reach a consensus about these values in each dimension. If there are different opinions amongst these software engineers, the final value assigned must be the average of the values assigned in the individual judgement.

The objective evaluation converts the qualitative values of the properties added to Table 1 into quantitative values, according to Table 6. In the course of evaluating comparison properties, the description of each property indicates the order of preference for qualitative values. Thus, a quantitative value equal to 1 corresponds to a lower qualitative value in the order of preference. The highest qualitative value in the order of preference has a quantitative value equal to 10 when there are two qualitative values for the property, and a quantitative value equal to 100 when there are three. In our case, these quantitative values, whether equal to 10 or 100, were intentionally attributed, so that the objective criteria prevailed over the subjective criteria.

Table 5. Template for evaluating the competence degree.

	Integration Platforms		
Dimension	*Platform 1*	...	*Platform n*
Message Processing	*	...	*
Hotspot Detection	*	...	*
Fairness Execution	*	...	*

Table 6. Correspondence of qualitative and quantitative values.

		Value	
Dimension	Property	Qualitative	Quantitative
Message Processing	Thread pool creation	global	1
		local	10
	Message store	in-memory	1
		hybrid	10
	Thread pool configuration	static	1
		dynamic	10
Hotspot Detection	Detection stage	design time	1
		run-time	10
	Abstraction level	per group	1
		per task	10
	Pattern identification	no	1
		yes	10
Fairness Execution	Execution model	process-based	1
		task-based	10
		hybrid	100
	Scheduling policy	fifo	1
		priority	10
		mapping	100
	Throttling controller	no	1
		yes	10

Table 7. Template for quantitative values.

		Integration Platforms		
Dimension	Property	*Platform 1*	···	*Platform n*
Message Processing	Thread pool creation	*	···	*
	Message store	*	···	*
	Thread pool configuration	*	···	*
Total		$\sum *$	···	$\sum *$
Hotspot Detection	Abstraction level	*	···	*
	Detection stage	*	···	*
	Pattern identification	*	···	*
Total		$\sum *$	···	$\sum *$
Fairness Execution	Execution model	*	···	*
	Scheduling policy	*	···	*
	Throttling controller	*	···	*
Total		$\sum *$	···	$\sum *$

Table 7 presents a template that can be used by software engineers to register the values of the competence scores of the integration platforms, obtained by the objective evaluation. The columns in the table represent the integration platforms alternatives and the lines represent the properties in the respective dimensions. At the end of each dimension, one line represents the total amount for the quantitative values in the dimension. The correspondence of qualitative and quantitative values was based on our experience in dealing with real-world integration problems by focusing on the performance of the platforms.

3.2.3. Score computation

This activity consists of the processing of data produced during the aforementioned activities and the calculation of the total score for each integration platform in each dimension. This calculation is made by merging subjective or objective criteria regarding the competence degree of the integration platforms, as well as the relevance level of each dimension. At the end of this activity, each integration platform has a score according to its competence degree, as well as a partial score and a total score. The score is an independent numeric value that measures the competence degree of the platform in a dimension. The partial score is a numeric value score for a platform in a dimension, which depends on the weight of the dimension. The total score is a numeric value corresponding to the accumulated partial score value of the three dimensions.

The subjective criteria are the judgements made by software engineers about the competence of each integration platform in finding a dimension. Table 5 has to be used to determine the final qualitative values to show that a consensus judgement has been reached. Equation (1) represents subjective criteria, which are equivalent to qualitative values, where *i* represents each integration platform and *k* represents each dimension.

$$subjective\_criteria_{i_k} \equiv qualitative\_value_{i_k} \tag{1}$$

The objective criteria involve the numeric conversion of the qualitative values of the properties into quantitative values. Table 7 summarises them by the sum of the quantitative values of properties in each dimension. According to Equation (2), objective criteria are the summation of the quantitative values by dimension, where *i* represents each integration platform, *k* represents each dimension and *j* represents each quantitative value.

$$objective\_criteria_{i_k} = \sum_{j=1}^{9} quantitative\_value_{i_k}[j] \tag{2}$$

The score achieved in terms of the competence degree of an integration platform (a_{i_k}) is the sum of the objective and subjective criteria, cf. Equation (3). In this equation, *i* represents each integration platform and *k* represents each dimension. The competence degree of a platform involves a score indicating that an integration platform has been realised in each dimension, regardless of the relevance level of the dimension. It is important to observe that the maximum quantitative value is greater than the maximum value of the scale of competence. This ensures that an integration platform with a high summation of the quantitative values is better ranked, although the value assigned to the subjective judgement of the software engineers is not sufficiently high. Thus, objective criteria prevail over subjective criteria.

$$a_{i_k} = objective\_criteria_{i_k} + subjective\_criteria_{i_k} \tag{3}$$

The partial score for an integration platform is the multiplication of the score achieved in the competence degree of a platform in meeting a dimension (a_{i_k}) by the weight of the respective dimension (ω_k), as set out in Equation (4). In this equation, *i* represents each integration platform and *k* represents each dimension.

$$partial\_score_{i_k} = a_{i_k} \cdot \omega_k \tag{4}$$

The total score for an integration platform is the summation of the partial scores in the three dimensions, i.e., the multiplication of the score achieved in terms of the competence degree of a platform (a_{i_k}) by the weight of the respective dimension (ω_k), as set out in Equation (5). In this equation, *i* represents each integration platform and *k* represents each dimension. *k* assumes values from 1 to 3, which correspond to the dimensions of message processing, hotspot detection and fairness execution, respectively. Lastly, an integration platform *i* has a competence degree score, a partial score and a total score.

$$total\_score[i] = \sum_{k=1}^{3} a_{i_k} \cdot \omega_k \tag{5}$$

3.2.4. Pairwise comparison

According to the psychological field, in order to arrive at a consensus, it is easier and more accurate to express an opinion about two alternatives than about all the alternatives at the same time (Ishizaka and Labib 2011). Pairwise comparisons are central to our methodology, as they allow software engineers to focus on a small and well-defined action involving the comparison of two elements in a given context. Software engineers use Table 8 in order to evaluate the integration platforms, attributing the value in a scale from 1 to 9.

Table 8. The fundamental Saaty scale.

Scale*	Definition	Explanation
1	Equal importance	Two properties contribute equally to the objective
3	Moderate importance of one over another	Experience and judgement strongly favour one property over another
5	Essential or strong importance	Experience and judgement strongly favour one property over another
7	Very strong importance	A property is strongly favoured and its dominance demonstrated in practice
9	Extreme importance	The evidence favouring one property over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between the two adjacent judgements	When compromise is needed
Reciprocals	If property *i* has one of the above numbers assigned to it when compared with property *j*, then *j* has a reciprocal value when compared with *i*	
Rationales	Ratios arising from the scale	If consistency were to be forced by obtaining *n* numerical values to span the matrix

*Intensity of importance on an absolute scale

Each pair of platforms is compared by judging how many times a platform is more competent than another to meet each dimension and to register the values of the judgement, in line with the template in Table 9. A value of 1 is given when integration platforms are equivalent or equal, i.e., both integration platforms have the same competence to meet the evaluated dimension. A value of 2 means that an integration platform is twice as competent than another. A value of 3 means that an integration platform is three times more competent than another. This continues successively up to value of 9, when an integration platform is nine times better than another. The paired judgement is a relative value (p_{ij}) or a quotient of two values (a/b) in which the value a is attributed to platform i, while the value b is attributed to platform j. The results of pairwise comparison for every integration platform are organised into positive reciprocal n x n matrix $P = (p_{ij})$, as shown in Equation (6), where n is the number of compared integration platforms. Pairwise comparisons are supported by the *TransparentChoice* software, which checks the consistency of the comparisons, points out potential errors and is able to forecast values of comparisons, based on the previously approximated results from the evaluation process, before making all comparisons.

Table 9. Template to summarise pairwise comparison.

		Integration Platforms		
Dimension		*Platform 1*	⋯	*Platform n*
Message Processing	***Platform 1***	1	⋯	*
	⋮	*	⋱	*
	Platform n	*	⋯	1
Hotspot Detection	***Platform 1***	1	⋯	*
	⋮	*	⋱	*
	Platform n	*	⋯	1
Fairness Execution	***Platform 1***	1	⋯	*
	⋮	*	⋱	*
	Platform n	*	⋯	1

$$P = \begin{pmatrix} 1 & p_{12} & \cdots & p_{1i} & \cdots & p_{1n} \\ p_{21} & 1 & \cdots & p_{2i} & \cdots & p_{2n} \\ \vdots & \ddots & \cdots & \vdots & \cdots & \vdots \\ p_{i1} & p_{i2} & \ddots & 1 & \cdots & p_{in} \\ \vdots & \vdots & \cdots & \ddots & \cdots & \vdots \\ p_{(n-1)1} & p_{(n-1)2} & \cdots & \vdots & 1 & p_{(n-1)n} \\ p_{n1} & p_{n2} & \cdots & p_{ni} & \cdots & 1 \end{pmatrix} \quad (6)$$

3.3. Data visualisation and interpretation

In this step, results are visualised and analysed according to the relevance level of the dimensions and the competence of the integration platforms. The relevance level of a dimension is the factor that most affects the ranking of an integration platform by making it better or worse. The consistency of the judgements should be checked according to the following points:

- The best overall position in the ranking of the integration platforms must be occupied by the platform that has the highest quantitative values for properties, since these values influence the score achieved in evaluating the competence degree of the integration platforms and, consequently, the total score.
- The best overall position in the ranking of the integration platforms must be occupied by the platform that best meets the dimension of a greater weight, according to Table 3.
- The best ranking of the integration platforms must be consistent with the pairwise comparison. Thus, if *Platform 1* is better than *Platform 2*, and *Platform 2* is better than *Platform 3, Platform 1* is also better than *Platform 3*.

The AHP method's mathematical base takes care of the consistency checking. A software tool can be used to perform consistency tests; thus, if there is any inconsistency in the judgements made by software engineers, the tool will warn them and the entire process should be repeated until all the inconsistencies have been solved, including the weight assignment, the rating scales and the pairwise comparisons. According to Sugden (1985), inconsistencies can be caused by clerical errors, incomplete model structure or by psychological reasons Some tools offer solutions, such as PriEsT that offers Pareto-optimal solutions based on multiobjective optimisation (Siraj, Mikhailov, and Keane 2015).

The ranking of the integration platforms can be presented as shown in the template in Figure 3. According to Kosara (2016), bar charts work well for classification and comparison and are easily understood and recognised across language barriers. This figure depicts the evaluation of each integration platform according to each dimension, and in general, as a partial score and total score, respectively. In this chart, each bar represents an integration platform. The width of a bar represents the value of the partial score for each dimension. Dark grey represents the partial score for the message processing dimension, light grey represents the partial score of the hotspot detection dimension, and white represents the partial score of the fairness execution dimension. The width of the entire

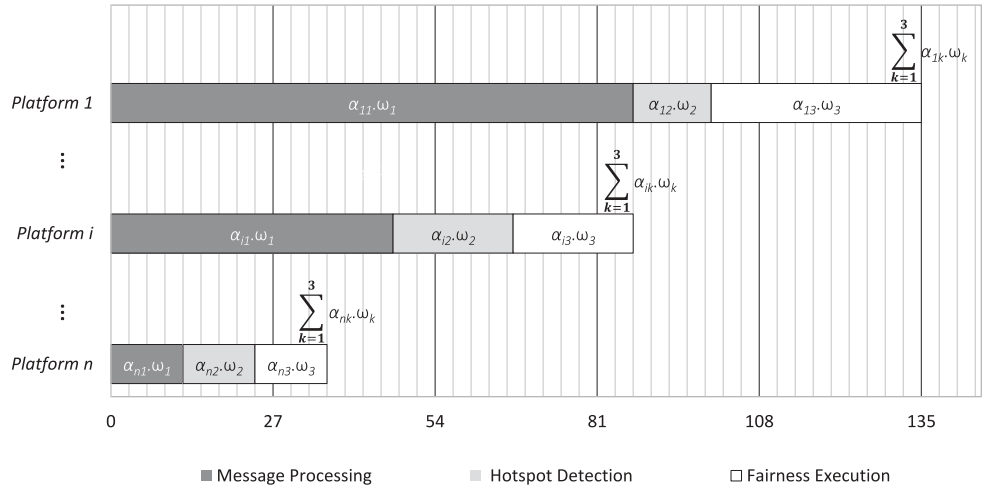

Figure 3. Template to present the ranking of the integration platforms.

bar represents the total score of an integration platform. The widest dark grey bar represents the best integration platform in terms of the message processing dimension. The widest light grey bar represents the best integration platform in terms of the hotspot detection dimension. The widest white bar represents the best integration platform in terms of the fairness execution dimension. The widest bar represents the best integration platform in general, i.e., in terms of all the dimensions. It is important to emphasise that the result could be different for the same set of integration platforms if the levels of relevance for the dimensions reported in Table 4 change.

4. Sample application

In this section, we demonstrate our methodology by carrying out a performance evaluation of five open-source state-of-the-art integration platforms. The chosen platforms were Spring Integration (Fisher et al. 2012), Camel (Ibsen and Anstey 2010), Fuse (Russell and Cohn 2012), Petals (Surhone, Timpledon, and Marseken 2010), and Guaraná (Frantz, Corchuelo, and Molina-Jiménez 2012). However, our methodology is applicable to any set of integration platforms that support the integration patterns (Hohpe and Woolf 2004) and follow the pipe-and-filter architectural style (Alexander, Ishikawa, and Silvertein 1977).

4.1. Evaluating comparison properties

The integration platforms were evaluated by comparing the properties of each dimension. The evaluation was done based on the findings of a multi-vocal literature review (Garousi, Felderer, and Mäntylä 2018) of publicly available documentation and source codes on websites, as well as books and articles. Table 10 presents the values found for every property in each of the three dimensions.

Table 10. Summary of the values of comparison properties.

		Integration Platforms				
Dimension	Property	*Spring Integration*	*Camel*	*Fuse*	*Petals*	*Guaraná*
Message Processing	Thread pool creation	local	local	global	global	global
	Message store	hybrid	hybrid	hybrid	in-memory	in-memory
	Thread pool configuration	static	static	static	static	static
Hotspot Detection	Detection stage	design time	run-time	design time	design time	design time
	Abstraction level	per group	per group	per group	per group	per task
	Pattern identification	no	no	no	no	no
Fairness Execution	Execution model	process-based	process-based	process-based	process-based	task-based
	Scheduling policy	fifo	fifo	fifo	fifo	fifo
	Throttling controller	yes	yes	no	no	no

Regarding the message processing dimension, Fuse, Petals and Guaraná have a global thread pool that executes all tasks in the integration solution. Spring Integration and Camel allow for the creation of local thread pools that can be dedicated to a task or a group of tasks in an integration solution. Run-time systems have seen advances in storing messages in the communication channels of an integration solution. Disk storage capacity facilitates the handling of big data (Chen, Mao, and Liu 2014), i.e., larger data in size or volume. Usually, disk storage can increase the total time for message execution, meaning that it can be used in scenarios in which this type of storage is really needed. Spring Integration, Camel and Fuse can store data in their memory and adopt strategies to deal with other storage types, such as in a file or on a database. In contrast, Petals and Guaraná are only able to store messages in their memory. Regarding thread pool configuration, no platform offers strategies to dynamically tune the size of the thread pool according to the demand of the execution of an integration solution. This ability allows run-time systems to efficiently deal with message processing peaks by assigning threads to highly demanded tasks, as well as releasing threads when the workload is lower.

The success of hotspot detection in the design phase still depends on the expertise of the software engineer. In the detection stage, every platform receives information that indicates the presence of hotspots in the execution of an integration solution, but only Camel is able to detect them during run-time. Regarding abstraction level of detection, every run-time system, except for Guaraná, is able to detect hotspots at the group level. In Guaraná, it is possible to specify tasks to be observed. As for the identification of patterns, none of the run-time systems is endowed with this property. Thus, the identification of such patterns depends on the quality of the information provided by the monitoring mechanisms, and on the experience of the software engineer. Most of the integration platforms use some kind of monitoring mechanism, which provides information to help with detecting bottlenecks, but they do not have any automated mechanism.

Regarding the fairness execution dimension, the majority of run-time systems adopt a process-based execution model, except for Guaraná, which adopts the task-based execution model. In Camel, the asynchronous approach uses staged event-driven architecture that encompasses a design philosophy for building more manageable multi-thread integration solutions, in which it is possible to allocate a thread pool to tasks in

a blocking queue. As for a task scheduling policy, every run-time system uses the first-in-first-out approach, while none of them allows tasks to have an associated priority to influence their scheduling, nor uses any mapping based on an exact method or optimisation. The throttling controller could help ensure a fairer execution when the rate of incoming messages is high, but only in Spring Integration and Camel. Further, it is possible to arrange tasks to actively call a thread at regular time intervals.

4.2. Ranking construction

In this section, we discuss the process of ranking the integration platforms by carrying out the activities in our methodology. The dimensions were prioritised, the platforms competences were evaluated by objective or subjective criteria, the scores were calculated and the pairwise comparisons were performed.

4.2.1. Dimension relevance

In this activity, we assigned the weight corresponding to the relevance level to each dimension, cf. Table 3. Message processing was considered as an *essential* dimension. Thus, high values allocated to the properties of the Message processing indicate that the platform is efficient. Hotspot detection was considered as a *desirable* dimension. Therefore, this dimension adds elements that enhance bottleneck detection, but they are not essential. The Fairness Execution was considered an *important* dimension. Therefore, the values of the properties of the Fairness execution have a medium relevance level. These prioritisation are shown in Table 11.

4.2.2. Platform competence

In this activity, the competence of each integration platform was quantified by subjective and objective evaluations. The subjective evaluation was performed with the help of the Saaty scale, resulting in Table 12. The objective evaluation was performed by converting the qualitative values of the properties into quantitative values, using Table 6 and resulting in Table 13.

4.2.3. Score computation

The score for each integration platforms is calculated by merging subjective and objective criteria regarding the competence of the integration platforms, as well as the weight of each dimension. At the end of this activity, each integration platform achieves an individual score (a_{i_k}), a partial score $(partial\_score_{i_k})$ and a total score $(total\_score[i])$. The index i assumes values from 1 to 5, corresponding to Spring Integration, Camel, Fuse, Petals or Guaraná, while the index k assumes values from 1 to 3, corresponding to the dimensions of message processing, hotspot detection and fairness execution, respectively.

The score computation starts with an evaluation of subjective criteria, according to Table 12, which provides the resulting values of our judgement concerning the competence of each integration platform to meet every dimension. The objective criteria are determined according to Table 13, where the sum of the quantitative values of the properties is on the lines entitled «Total». Equation (7) presents matrices with these values, where the line index corresponds to every integration platform, and the column index corresponds to every dimension.

$$subjective\_criteria = \begin{bmatrix} 8 & 1 & 4 \\ 8 & 4 & 4 \\ 4 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 4 & 4 \end{bmatrix} \quad objective\_criteria = \begin{bmatrix} 21 & 3 & 12 \\ 21 & 12 & 12 \\ 12 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 12 & 12 \end{bmatrix} \tag{7}$$

The score achieved in relation to the competence degree of an integration platform is equal to the sum of the objective and the subjective criteria, cf. Equation (8). In this matrix, the elements correspond to the achieved score, the line index corresponds to every integration platform, and the column index corresponds to every dimension.

$$a = \overbrace{\begin{bmatrix} 8 & 1 & 4 \\ 8 & 4 & 4 \\ 4 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 4 & 4 \end{bmatrix}}^{subjective\_criteria} + \overbrace{\begin{bmatrix} 21 & 3 & 12 \\ 21 & 12 & 12 \\ 12 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 12 & 12 \end{bmatrix}}^{objective\_criteria} = \begin{bmatrix} 29 & 4 & 16 \\ 29 & 16 & 16 \\ 16 & 4 & 4 \\ 4 & 4 & 4 \\ 4 & 16 & 16 \end{bmatrix} \tag{8}$$

The partial score is equal to the score achieved in relation to the competence degree multiplied by the weight of the respective dimension, cf. Equation (9). In this matrix, the elements correspond to the achieved score in the dimension, the line index corresponds to every integration platform, and the column index corresponds to every dimension.

$$partial\_score = \overbrace{\begin{bmatrix} 29 & 4 & 16 \\ 29 & 16 & 16 \\ 16 & 4 & 4 \\ 4 & 4 & 4 \\ 4 & 16 & 16 \end{bmatrix}}^{a} \times \overbrace{\begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}}^{\omega} = \begin{bmatrix} 87 & 4 & 32 \\ 87 & 16 & 32 \\ 48 & 4 & 8 \\ 12 & 4 & 8 \\ 12 & 16 & 32 \end{bmatrix} \tag{9}$$

The total score of every integration platform is equal to the sum of the partial scores, i.e., the sum of the elements of all columns in the matrix 9. Equation (10) presents a matrix with these values, where the line index corresponds to every integration platform.

$$total\_score = \overbrace{\begin{bmatrix} 87 & + & 4 & + & 32 \\ 87 & + & 16 & + & 32 \\ 48 & + & 4 & + & 8 \\ 12 & + & 4 & + & 8 \\ 12 & + & 16 & + & 32 \end{bmatrix}}^{\sum_{k=1}^{3} partial\_score_{i_k}} = \begin{bmatrix} 123 \\ 135 \\ 60 \\ 24 \\ 60 \end{bmatrix} \tag{10}$$

4.2.4. Pairwise comparison

In this activity, we compared each pair of integration platforms and assessed how many times a platform is better than another, using the Saaty scale. They were compared two by two for each dimension, taking into account the values of Table 12. When a platform was compared with itself, the value of 1 was assigned. The pairwise comparison results are presented in Table 14.

In the message processing dimension, Spring Integration and Camel were considered to be eight times better than Petals and Guaraná, and twice as good as Fuse. In this dimension,

Petals and Guaraná were considered to be four times better than Fuse, Spring Integration was considered to be equivalent to Camel, and Petals was considered to be equivalent to Guaraná. In the hotspot detection dimension, Camel and Guaraná were considered to be equivalent to each other, but four times better than Spring Integration, Fuse and Petals. In this dimension, Spring Integration, Fuse and Petals were considered to be equivalent to each other. In the fairness execution dimension, Spring Integration, Camel and Guaraná were considered as equivalent to each other, but four times better than Fuse and Petals. In this dimension, Fuse and Petals were considered as equivalent to each other.

We propose the use of a tool for making judgements and verifying the consistency of the outcomes, in order to simplify the ranking construction process for software engineers. However, we include an explanation of the classic approach in the Appendices 6, in order to clarify how consistency is verified. For more details, we suggest Saaty (1990), Saaty (2008), Ishizaka and Labib (2011) and Franek and Kresta (2014).

4.3. Data visualisation and interpretation

In this section, we analyse the results of the integration platforms' evaluation using a graphical representation. The platforms' total scores are expressed analytically by a summation of the score achieved in relation to the competence degree in every dimension multiplied by the weight of this dimension, cf. Equation (11). Equations are in total scores' descending order. The platforms' total scores are expressed graphically by Figure 4, that shows this raking by discriminating between the total and the partial score in each of the three dimensions.

$$\begin{aligned}
\textit{Camel}: &= 29\cdot 3 + 16\cdot 1 + 16\cdot 2 = 135\\
\textit{Spring Integration}: &= 29\cdot 3 + 4\cdot 1 + 16\cdot 2 = 123\\
\textit{Guaraná}: &= 4\cdot 3 + 16\cdot 1 + 16\cdot 2 = 60\\
\textit{Fuse}: &= 16\cdot 3 + 4\cdot 1 + 4\cdot 2 = 60\\
\textit{Petals}: &= 4\cdot 3 + 4\cdot 1 + 4\cdot 2 = 24
\end{aligned} \tag{11}$$

The ranking indicates that the Camel run-time system offers the best performance amongst them, with a score of 135, followed by Spring Integration, Fuse, Guaraná and Petals respectively with a score of 123, 60, 60, and 24. Camel and Spring Integration were the best platforms in the message processing dimension, with a score of 87, while Fuse scored 48, and Petals and Guaraná scored 12. The relevance level of this dimension was considered essential in order to realise good performance. Camel and Guaraná were the best platforms in the hotspot detection dimension, with a score of 16, whereas Spring Integration, Fuse and Petals scored 4. The relevance level of this dimension was considered desirable. Spring Integration, Camel and Guaraná were the best platforms in the fairness execution dimension, with a score 32, while Petals and Fuse scored 8. The relevance level of this dimension was considered important.

The message processing dimension is best served by Spring Integration, Camel, Fuse and Petals. The second dimension best served by these platforms is fairness execution. Meanwhile, the fairness execution dimension is best served by Guaraná, and the second dimension best served by this platform is message processing. The hotspot detection dimension is the dimension that is least served by the five platforms. Camel, Fuse, Petals, and Guaraná had the same competence degree score in the hotspot detection and

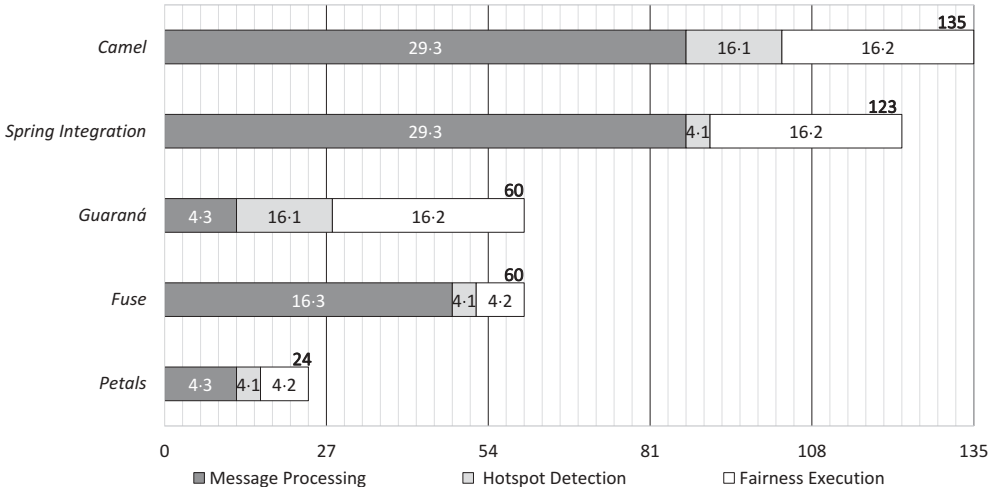

Figure 4. Ranking of integration platforms.

fairness execution dimensions. However, the relevance level of the fairness execution dimension was considered higher than that of hotspot detection; thus, the partial scores between these dimensions were different for these dimensions.

An important point to be observed is that the high partial score in the message processing dimension leveraged the total score for Spring Integration, Camel and Fuse, because this dimension was considered to be more relevant than the others, with a relevance-level weight equal to 3. Likewise, the high partial score in the fairness execution dimension increased the total score for Spring Integration, Camel and Guaraná, because this dimension was considered more relevant than that of hotspot detection.

5. Conclusions

New trends, such as cloud computing and big data, have provided several opportunities for companies to leverage their business processes. However, their software ecosystems can become more heterogeneous as a consequence. Hence, companies have started to become concerned about the performance of their integration platforms in terms of providing synchronisation and data exchange efficiently across their applications. The run-time system is one of the elements of a platform that most influences performance, because it is responsible for running integration solutions. Thus, it should have more weight in companies' decision-making process for an integration platform. As there are many alternatives in the market, software engineers need support in the evaluation and selection of platforms. In this article, we presented a methodology to compare and rank integration platforms by focusing on the performance of their run-time systems.

The proposed methodology is based on the AHP, one of whose main benefits is the modelling of a complex decision problem in a hierarchical structure, divided into multiple levels. In our case, there are three hierarchical levels. At the top of this hierarchy is the main goal, i.e., choosing the best performance. At the second level are the decision criteria, i.e, message processing, hotspot detection and fairness execution dimensions. The first dimension

focuses on the efficiency of message processing within an integration solution, while the second dimension focuses on the detection of bottlenecks, and the third dimension focuses on the balanced distribution of computational resources for the tasks within an integration solution. At the lowest level are the alternatives, i.e., the integration platforms.

There are three well-defined steps in the proposed methodology: evaluating comparison properties, ranking construction, and data visualisation and interpretation. When evaluating comparison properties, values are attributed to performance properties. The ranking construction step determines the relevance level of the dimensions and calculates the scores for the integration platforms. The data visualisation and interpretation step ranks the platforms. The proposed methodology applied objective and subjective criteria to evaluate integration platforms. The objective evaluation considered the performance property values of the dimensions and the subjective evaluation took the preferences of experts into account. The objective criteria prevailed over the subjective criteria, ensuring that an integration platform with more objective values is better ranked, even though the subjective judgement may not be as persuasive.

To confirm the feasibility of the proposed methodology, we evaluated five integration platforms: Camel, Spring Integration, Fuse, Petals and Guaraná. In this evaluation, the priority order for the dimensions was message processing (essential), fairness execution (important) importance, and hotspot detection (desirable). In turn, we ranked the overall performance of the platforms. Camel was the best amongst them, followed by Spring Integration, Fuse, Guaraná and Petals, in descending order. It was also possible to observe the results for each of the dimensions separately. Regarding the message processing dimension, Camel and Spring Integration were equivalent and obtained the highest score; regarding hotspot detection, Camel was the best; and regarding the fairness execution dimension, Camel, Spring Integration and Guaraná were equivalent and achieved the highest score.

Notes

1. https://www.transparentchoice.com/ahp-software.
2. https://www.expertchoice.com.
3. https://www.superdecisions.com.
4. https://bpmsg.com/academic/ahp.php.
5. https://github.com/czekster/Decisor/releases.
6. https://sourceforge.net/projects/priority/.
7. The spectral radius of a square matrix is the largest eigenvalue in terms of absolute value.

Acknowledgments

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) under Grant Numbers 73318345415 and 88881.119518/2016-01, as well as the Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) under Grant Number 17/2551-0001206-2. We would like to thank Dr Rafael Corchuelo and Dr Inma Hernández from the University of Seville (Spain) and Ms Elizabeth Thornton Rush from Pennsylvania State University (United States) for their helpful comments in earlier versions of this article.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior [73318345415, 88881.119518/2016-01]; Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul [17/2551-0001206-2].

ORCID

Rafael Z. Frantz http://orcid.org/0000-0003-3740-7560
Fabricia Roos-Frantz http://orcid.org/0000-0001-9514-6560

References

Ahmad, S. 2016. "Load Balancing in Distributed Framework for Frequency Based Thread Pools." *Computational Ecology and Software* 6 (4): 150–164.

Aldin, L., and S. de Cesare. 2011. "A Literature Review on Business Process Modelling: New Frontiers of Reusability." *Enterprise Information Systems* 5 (3): 359–383. doi:10.1080/17517575.2011.557443.

Alexander, C., S. Ishikawa, and M. Silvertein. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford: Oxford University Press.

Anwar, N., and H. Deng. 2018. "Elastic Scheduling of Scientific Workflows under Deadline Constraints in Cloud Computing Environments." *Future Internet* 10 (1): 1–23. doi:10.3390/fi10010005.

Balko, S., and A. Barros. 2015. "In-Memory Business Process Management." International Conference on Enterprise Distributed Object Computing (EDOC), Adelaide, Australia, 74–83.

Botta, A., W. de Donato, V. Persico, and P. Antonio. 2016. "Integration of Cloud Computing and Internet of Things: A Survey." *Future Generation Computer Systems* 56: 684–700. doi:10.1016/j.future.2015.09.021.

Brahmi, Z., and C. Gharbi. 2014. "Temporal Reconfiguration-Based Orchestration Engine in the Cloud Computing." In *International Conference on Business Information Systems (ICBIS)*, 73–85.

Chen, M., S. Mao, and Y. Liu. 2014. "Big Data: A Survey." *Mobile Networks and Applications* 19: 171–209. doi:10.1007/s11036-013-0489-0.

Chen, W., and E. Deelman. 2011. "Workflow Overhead Analysis and Optimizations." Workshop on Workflows in Support of Large-scale Science (WORKS), 11–20.

Corchuelo, R., R. Z. Frantz, and J. Gonzáles. 2008. "Una Comparación De ESBs Desde La Perspectivade La Integración De Aplicaciones." Jornadas de Ingeniería del Software y Bases de Datos (JISBD), 403–408.

Czekster, R. M., H. J. De Carvalho, G. Z. Kessler, L. M. Kipper, and T. Webber. 2019. "Decisor: A Software Tool to Drive Complex Decisions with Analytic Hierarchy Process." *International Journal of Information Technology & Decision Making* 18 (1): 65–86. doi:10.1142/S0219622018500360.

Ebert, N., and K. Weber. 2016. "Integration Platform as a Service in Der Praxis: Eine Bestandsaufnahme." In *Multikonferenz Wirtschaftsinformatik (MKWI)*, 1675–1685.

Ebert, N., K. Weber, and S. Koruna. 2017. "Integration Platform as a Service." *Business & Information Systems Engineering* 59: 375–379. doi:10.1007/s12599-017-0486-0.

Fisher, M., J. Partner, M. Bogoevice, and I. Fuld. 2012. *Spring Integration in Action*. Manning Publications.

Franek, J., and A. Kresta. 2014. "Judgment Scales and Consistency Measure in AHP." *Procedia Economics and Finance* 12: 164–173. doi:10.1016/S2212-5671(14)00332-3.

Frantz, R. Z., R. Corchuelo, and J. L. Arjona. 2011. "An Efficient Orchestration Engine for the Cloud." IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 711–716.

Frantz, R. Z., R. Corchuelo, and C. Molina-Jimenéz. 2012. "A Proposal to Detect Errors in Enterprise Application Integration Solutions." *Journal of Systems and Software* 85 (3): 480–497. doi:10.1016/j.jss.2011.10.048.

Frantz, R. Z., R. Corchuelo, and F. Roos-Frantz. 2016. "On the Design of a Maintainable Software Development Kit to Implement Integration Solutions." *Journal of Systems and Software* 111: 89–104. doi:10.1016/j.jss.2015.08.044.

Freire, D. L., R. Z. Frantz, and F. Roos-Frantz. 2019. "Ranking Enterprise Application Integration Platforms from a Performance Perspective: An Experience Report." *Software: Practice and Experience* 49 (5): 921–941.

Freire, D. L., R. Z. Frantz, F. Roos-Frantz, and S. Sawicki. 2019. "Survey on the Run-Time Systems of Enterprise Application Integration Platforms Focusing on Performance." *Software: Practice and Experience* 49 (3): 341–360.

Garousi, V., M. Felderer, and M. V. Mäntylä. 2018. "Guidelines for Including Grey Literature and Conducting Multivocal Literature Reviews in Software Engineering." *Information and Software Technology* 1–22.

Gleyzer, G., and J. Howes. 2017. "System and Method for Supporting Dynamic Thread Pool Sizing in a Distributed Data Grid." US Patent 9,547,521 B2.

Guttridge, K., M. Pezzini, E. Golluscio, E. Thoo, K. Iijima, and M. Wilcox. 2017. "Magic Quadrant for Enterprise Integration Platform as a Service 2017." *Technical Report*. Stamford, CT: Gartner, Inc.

Harman, M., K. Lakhotia, J. Singer, D. R. White, and S. Yoo. 2013. "Cloud Engineering Is Search Based Software Engineering Too." *Journal of Systems and Software* 86 (9): 2225–2241. doi:10.1016/j.jss.2012.10.027.

Hohpe, G., and B. Woolf. 2004. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA: Addison-Wesley Professional.

Ibsen, C., and J. Anstey. 2010. *Camel in Action*. Stamford, CT: Manning Publications.

Ishizaka, A., and A. Labib. 2011. "Review of the Main Developments in the Analytic Hierarchy Process." *Expert Systems with Applications* 38: 14336–14345.

Kawaguchi, A., and H. Yamada. 2005. "Methodology of Performance Evaluation of Integrated Service Systems with Timeout Control Scheme." Asia-Pacific Network Operations and Management Symposium (APNOMS), Okinawa, Japan, 235–246.

Khoumbati, K., M. Themistocleos, and Z. Irani. 2006. "Evaluating the Adoption of Enterprise Application Integration in Health-Care Organizations." *Journal of Management Information Systems* 22: 69–108. doi:10.2753/MIS0742-1222220404.

Kosara, R. 2016. "Presentation-Oriented Visualization Techniques." *IEEE Computer Graphics and Applications* 36 (1): 80–85. doi:10.1109/MCG.2016.2.

Linthicum, D. S. 2017. "Cloud Computing Changes Data Integration Forever: What's Needed Right Now." *IEEE Cloud Computing* 4 (3): 50–53. doi:10.1109/MCC.2017.47.

Manikas, K. 2016. "Revisiting Software Ecosystems Research: A Longitudinal Literature Study." *Journal of Systems and Software* 117: 84–103. doi:10.1016/j.jss.2016.02.003.

More, V. R., and M. M. Bartere. 2013. "Enterprise Integration Using Boomi Tool." *International Journal of Advanced Information Science and Technology* 11: 18–22.

Perron, O. 1907. "Grundlagen Für Eine Theorie Des Jacobischen Kettenbruchalgorithmus." *Mathematische Annalen* 64 (1): 1–76. doi:10.1007/BF01449880.

Pfaff, M., and H. Krcmar. 2018. "A Web-Based System Architecture for Ontology-Based Data Integration in the Domain of IT Benchmarking." *Enterprise Information Systems* 12 (3): 236–258. doi:10.1080/17517575.2017.1329552.

Ritter, D., J. Dann, N. May, and S. Rinderle-Ma. 2017. "Hardware Accelerated Application Integration Processing: Industry Paper." International Conference on Distributed and Event-based Systems (DEBS), Barcelona, Spain, 215–226.

Ritter, D., N. May, and S. Rinderle-Ma. 2017. "Patterns for Emerging Application Integration Scenarios: A Survey." *Information Systems* 67: 36–57. doi:10.1016/j.is.2017.03.003.

Roos-Frantz, F., M. Binelo, R. Z. Frantz, S. Sawicki, and V. B. Fernandes. 2015. "Using Petri Nets to Enable the Simulation of Application Integration Solutions Conceptual Models." Conference on Enterprise Information Systems(ICEIS), 87–96.

Russell, J., and R. Cohn. 2012. *Fuse ESB*. Edited by F. P. Miller, A. F. Vandome, M. John. Riga, Latvia: VDM Publishing.

Saaty, T. L. 1990. "How to Make a Decision: The Analytic Hierarchy Process." *European Journal of Operational Research* 48: 9–26. doi:10.1016/0377-2217(90)90057-I.

Saaty, T. L. 2008. "Relative Measurement and Its Generalization in Decision Making Why Pairwise Comparisons are Central in Mathematics for the Measurement of Intangible Factors the Analytic Hierarchy/Network Process." *Revista De La Real Academia De Ciencias Exactas, Fisicas Y Naturales* 102: 251–318.
Santos, J., J. M. Sarriegi, and N. Serrano. 2008. "A Support Methodology for EAI and BPM Projects in SMEs." *Enterprise Information Systems* 2 (3): 275–286. doi:10.1080/17517570802262719.
Sharma, S. 2017. "Ovum Decision Matrix highlights the growing importance of iPaaS and API platforms in hybrid integration." *Technical Report*. London, UK: Ovum Consulting.
Siraj, S., L. Mikhailov, and J. A. Keane. 2015. "PriEsT: An Interactive Decision Support Tool to Estimate Priorities from Pairwise Comparison Judgments." *International Transactions in Operational Research* 22 (2): 217–235. doi:10.1111/itor.12054.
Sudarsanam, A., M. Srinivasan, and S. Panchanathan. 2004. "Resource Estimation and Task Scheduling for Multithreaded Reconfigurable Architectures." International Conference on Parallel and Distributed Systems (ICPADS), Newport Beach, CA, 323–330.
Sugden, R. 1985. "Why Be Consistent? A Critical Analysis of Consistency Requirements in Choice Theory." *Economica* 52 (206): 167–183. doi:10.2307/2554418.
Surhone, L. M., M. T. Timpledon, and S. F. Marseken. 2010. *Petals ESB*. Beau Bassin, Mauritius: Betascript Publishing.
Tan, S., H. K. B. Milhim, B. Chen, A. Schiffauerova, and Y. Zeng. 2011. "Enterprise Applications Integration Using Environment Based Design (EBD)." International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (ASME), Washington, DC, 1245–1255.
Traore, I., I. Woungang, A. A. E. S. Ahmed, and M. S. Obaidat. 2012. "Software Performance Modeling Using the UML: A Case Study." *Journal of Networks* 7 (1): 4–20. doi:10.4304/jnw.7.1.4-20.
Vaidya, O. S., and S. Kumar. 2006. "Analytic Hierarchy Process: An Overview of Applications." *European Journal of Operational Research* 169: 1–29. doi:10.1016/j.ejor.2004.04.028.
Varghese, B., and R. Buyya. 2018. "Next Generation Cloud Computing: New Trends and Research Directions." *Future Generation Computer Systems* 79: 849–861. doi:10.1016/j.future.2017.09.020.

Appendices

A. Classic Approach

In this section, we applied a classic approach to verify the consistency results. We used one of the simplest algebraic methods in order to demonstrate how it is applied using support tools. As the algebraic method is not feasible for high-dimension matrices, numeric methods are adopted, which have different stop criteria, precision degrees and a number of interaction constraints. The study of such methods exceeds the scope of the proposed methodology.

A.1. *Pairwise Dimensions Comparison*

According to Table 11, the decision matrix for the first level of the hierarchy structure is:

$$D = \begin{bmatrix} 1 & 3 & 3/2 \\ 1/3 & 1 & 1/2 \\ 2/3 & 2 & 1 \end{bmatrix} \tag{12}$$

First, it is necessary to calculate the eigenvectors and eigenvalues for decision matrices. We use characteristic equations for matrices. Equation (13) expresses the general formula for a characteristic equation.

$$\det(\mathrm{D} - \lambda_D \mathcal{I}) = {\lambda_D}^2 \cdot (\lambda_D - 3) = 0 \tag{13}$$

Table 11. Summary of dimension relevance.

Dimension	Essential	Important	Desirable
Message Processing	3	–	–
Hotspot Detection	–	–	1
Fairness Execution	–	2	–

Table 12. Competence degree.

	Integration Platforms				
Dimension	*Spring Integration*	*Camel*	*Fuse*	*Petals*	*Guaraná*
Message Processing	8	8	4	1	1
Hotspot Detection	1	4	1	1	4
Fairness Execution	4	4	1	1	4

Table 13. Quantitative values for the competence of each integration platform.

		Integration Platforms				
Dimension	Property	*Spring Integration*	*Camel*	*Fuse*	*Petals*	*Guaraná*
Message Processing	Thread pool creation	10	10	1	1	1
	Storage message	10	10	10	1	1
	Thread pool configuration	1	1	1	1	1
Total		21	21	12	3	3
Hotspot Detection	Detection stage	1	10	1	1	1
	Abstraction level	1	1	1	1	10
	Pattern identification	1	1	1	1	1
Total		3	12	3	3	12
Fairness Execution	Execution model	1	1	1	1	10
	Scheduling policy	1	1	1	1	1
	Throttling controller	10	10	1	1	1
Total		12	12	3	3	12

The solutions of the characteristic equations are $\lambda = 3$ with a multiplicity equal to 1 and $\lambda = 0$ with a multiplicity equal to 2. The theorem proposed by Perron (1907), a German mathematician, insists that:

> *A positive square matrix has an eigenvalue with a multiplicity equal to 1 in relation to its spectral radius,*[7] *although no eigenvalue is so large in terms of absolute value. In addition, there is an eigenvector on the right and an eigenvector on the left corresponding to the spectral value only with positive components.*

The theorem proves that, if $D = (d_{ij})$, $d_{ij} > 0$, $i, j = 1, 2, \ldots n$, D has a simple positive eigenvalue λ_{max}, known as the principal eigenvalue of D and $\lambda_{max} > |\lambda_k|$ for the remaining eigenvalues of D.

Thus, it is necessary to obtain the highest eigenvalue λ_{max} associated with the main eigenvector of the positive matrix. In this case, the eigenvalue is $\lambda_{max} = 3$. The priorities vector of the elements in the decision matrix is obtained by the right eigenvector method. This eigenvector is such that $DW = \lambda W$, thus:

Table 14. Summary of the pairwise comparison results.

Dimension		Integration Platforms				
		Spring Integration	*Camel*	*Fuse*	*Petals*	*Guaraná*
Message Processing	***Spring Integration***	1x	1x	2x	8x	8x
	Camel	1x	1x	2x	8x	8x
	Fuse	1/2x	1/2x	1x	4x	4x
	Petals	1/8x	1/8x	1/4x	1x	1x
	Guaraná	1/8x	1/8x	1/4x	1x	1x
Hotspot Detection	***Spring Integration***	1x	1/4x	1x	1x	1/4x
	Camel	4x	1x	4x	4x	1x
	Fuse	1x	1/4x	1x	1x	1/4x
	Petals	1x	1/4x	1x	1x	1/4x
	Guaraná	4x	1x	4x	4x	1x
Fairness Execution	***Spring Integration***	1x	1x	4x	4x	1x
	Camel	1x	1x	4x	4x	1x
	Fuse	1/4x	1/4x	1x	1x	1/4x
	Petals	1/4x	1/4x	1x	1x	1/4x
	Guaraná	1x	1x	4x	4x	1x

$$\overbrace{\begin{bmatrix} 1 & 3 & 3/2 \\ 1/3 & 1 & 1/2 \\ 2/3 & 2 & 1 \end{bmatrix}}^{D} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \overbrace{3}^{\lambda_{max}} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \tag{14}$$

Then,

Table A1. Saaty's random indices.

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.52	0.8	1.11	1.25	1.35	1.4	1.45	1.49

$$W_D = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 3w_2 \\ w_2 \\ 2w_2 \end{bmatrix} = w_2 \times \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \tag{15}$$

The consistency index was defined by Saaty (1990) as:

$$CI = (\lambda_{max} - n) \cdot (n - 1) \tag{16}$$

where λ_{max} is the largest eigenvalue associated with the main eigenvector of the positive matrix and n is the order of the matrix. Saaty proposes random indices (RI) for matrices with dimensions from 1 to 10, cf. Table A1. The consistency index (CI) is compared with the random indices to produce the consistency ratio (CR), where $CR = CI/RI$. If $CR < 0.1$, then judgements about the decision matrix are consistent; otherwise, there is some inconsistency and the judgements must be made again.

In our case, $\lambda_{max} = 3$ and $n = 3$, then:

$$CI = (\lambda_{max} - n) \cdot (n - 1) = (3 - 3) \cdot (3 - 1) = 0$$

$$CR = CI/RI = 0/1.11 = 0$$

Thus, as in the case of $CR < 0.1$, the judgements about the decision matrix are consistent.

A.2. *Pairwise Comparison of Platforms*

According to Table 14, there are three decision matrices for the second level of the hierarchy structure. In Equation (17), «A» refers to the message processing decision matrix, «B» refers to the hotspot detection decision matrix, and «C» refers to the fairness execution decision matrix. Equation (18) shows the characteristic equations for message processing, hotspot detection and fairness execution, respectively.

$$\overbrace{\begin{bmatrix} 1 & 1 & 2 & 8 & 8 \\ 1 & 1 & 2 & 8 & 8 \\ 1/2 & 1/2 & 1 & 4 & 4 \\ 1/8 & 1/8 & 1/4 & 1 & 1 \\ 1/8 & 1/8 & 1/4 & 1 & 1 \end{bmatrix}}^{A} \quad \overbrace{\begin{bmatrix} 1 & 1/4 & 1 & 1 & 1/4 \\ 4 & 1 & 4 & 4 & 1 \\ 1 & 1/4 & 1 & 1 & 1/4 \\ 1 & 1/4 & 1 & 1 & 1/4 \\ 4 & 1 & 4 & 4 & 1 \end{bmatrix}}^{B} \quad \overbrace{\begin{bmatrix} 1 & 1 & 4 & 4 & 1 \\ 1 & 1 & 4 & 4 & 1 \\ 1/4 & 1/4 & 1 & 1 & 1/4 \\ 1/4 & 1/4 & 1 & 1 & 1/4 \\ 1 & 1 & 4 & 4 & 1 \end{bmatrix}}^{C} \tag{17}$$

$$\begin{aligned} \det(\mathrm{A} - \lambda_A \mathcal{I}) &= {\lambda_A}^4 \cdot (\lambda_A - 5) = 0 \\ \det(\mathrm{B} - \lambda_B \mathcal{I}) &= {\lambda_B}^4 \cdot (\lambda_B - 5) = 0 \\ \det(\mathrm{C} - \lambda_C \mathcal{I}) &= {\lambda_C}^4 \cdot (\lambda_C - 5) = 0 \end{aligned} \tag{18}$$

The solutions of the characteristic equations are $\lambda = 5$ with a multiplicity equal to 1 and $\lambda = 0$ with a multiplicity equal to 4. According to Perron's theorem, $\lambda_{max} = 5$.

Equation (20) shows the message processing priorities vector, Equation (21) shows the hotspot detection priorities vector, and Equation (23) shows the fairness execution priorities vector.

$$\overbrace{\begin{bmatrix} 1 & 1 & 2 & 8 & 8 \\ 1 & 1 & 2 & 8 & 8 \\ 1/2 & 1/2 & 1 & 4 & 4 \\ 1/8 & 1/8 & 1/4 & 1 & 1 \\ 1/8 & 1/8 & 1/4 & 1 & 1 \end{bmatrix}}^{A} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \overbrace{5}^{\lambda_{max}} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \tag{19}$$

then,

$$W_A = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} w_2 \\ w_2 \\ w_2/2 \\ w_2/8 \\ w_2/8 \end{bmatrix} = w_2 \times \begin{bmatrix} 1 \\ 1 \\ 1/2 \\ 1/8 \\ 1/8 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0.5 \\ 0.125 \\ 0.125 \end{bmatrix} \tag{20}$$

$$\overbrace{\begin{bmatrix} 1 & 1/4 & 1 & 1 & 1/4 \\ 4 & 1 & 4 & 4 & 1 \\ 1 & 1/4 & 1 & 1 & 1/4 \\ 1 & 1/4 & 1 & 1 & 1/4 \\ 4 & 1 & 4 & 4 & 1 \end{bmatrix}}^{B} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \overbrace{5}^{\lambda_{max}} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \tag{21}$$

Then,

Table A2. Aggregation of priority vectors.

	Message Processing (3)	Hotspot Detection (1)	Fairness Execution (2)	
Spring Integration	1	0.25	1	5.25
Camel	1	1	1	6
Fuse	0.5	0.25	0.25	2.25
Petal	0.125	0.25	0.25	1.125
Guaraná	0.125	1	1	3.375

$$W_B = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} w_2/4 \\ w_2 \\ w_2/4 \\ w_2/4 \\ w_2 \end{bmatrix} = w_2 \times \begin{bmatrix} 1/4 \\ 1 \\ 1/4 \\ 1/4 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 1 \\ 0.25 \\ 0.25 \\ 1 \end{bmatrix} \tag{22}$$

$$\overbrace{\begin{bmatrix} 1 & 1 & 4 & 4 & 1 \\ 1 & 1 & 4 & 4 & 1 \\ 1/4 & 1/4 & 1 & 1 & 1/4 \\ 1/4 & 1/4 & 1 & 1 & 1/4 \\ 1 & 1 & 4 & 4 & 1 \end{bmatrix}}^{C} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \overbrace{5}^{\lambda_{max}} \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \tag{23}$$

then,

$$W_B = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} w_2 \\ w_2 \\ w_2/4 \\ w_2/4 \\ w_2 \end{bmatrix} = w_2 \times \begin{bmatrix} 1 \\ 1 \\ 1/4 \\ 1/4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0.25 \\ 0.25 \\ 1 \end{bmatrix} \tag{24}$$

In our case, $\lambda_{max} = 5$ and $n = 5$; thus, in three cases, the consistency index is:

$$CI = (\lambda_{max} - n) \cdot (n - 1) = (5 - 5) \cdot (5 - 1) = 0$$

$$CR = CI/RI = 0/1.11 = 0$$

In turn, as in the case of $CR < 0.1$, the judgements about the decision matrix are consistent.

A.3. *Results Aggregation*

After obtaining the priority vectors of the decision matrices, these vectors are aggregated to generate the final values of the platforms. We set out the competences of the platform with respect to each dimension in the matrix, multiply each column of vectors by the priority of the corresponding dimension, and add across each row, which results in the desired vector of the platforms in Table A2. In descending order of priority, the platforms are ranked as follows: Camel, Spring Integration, Guaraná, Fuse and Petal.