International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Unified Microservices Architecture for Enterprise HR Systems: Integrating Cognitive RPA and Cloud-Based HCM

Abhimanyu Kumar

Submitted: 02/05/2022 **Revised**: 15/06/2022 **Accepted**: 24/06/2022

Abstract— Cloud ERP systems are increasing the enterprise HRM systems limits to provide real-time insights into their workforce, scale and compliance whilst keeping costs under control. Today's platforms are still fragmented in architecture, having multiple scattered automation tools, cloud modules and legacy monolithic cores. This paper suggests an overall design of this integration based on microservices decomposition, cognitive RPA and HCM cloud platforms where they become a part of a streamlined and logical operating model. A 4 Layer Architecture is given which encompasses Service Decomposition, Automation Integration, Data Governance & Deployment Infrastructure. The proposed design got analyzed using quantitative performance models, formulas for scalability and empirical ones from literature. The results show that there is a unified architecture's deployment latency that can decrease by 40-60%, the process throughput for HR work that can increase by almost 35%, and the HR operational overhead linked to benefits administration can be reduced by 52% in comparison to traditional fragmented architectures. The paper also outlines parameters of migration, security restrictions and governance issues at an enterprise level.

Keywords-*Microservices, Robotic Process Automation, Human Capital Management, Cloud Architecture, Digital HRM, DevOps, CI/CD.*

I. INTRODUCTION

Multinational companies maintain thousands of HR records, distributed geographies and numerous HR processes. The infrastructure and systems that are supposed to aid this effort typically date back to a time when systems were deployed on-premise and the cycle for releases was once a year. They were not meant to be leveraged for Continuous change, API Integration or Machine Learning Based Decision Support.

There have been three separate events that have led to the possibility of creating better architecture. Cloud applications are now built with microservices, and the validated patterns of moving to the cloud and the tooling for microservices' deployments have grown. Armed with programmatic access through standardized APIs, cloud-based HCM platforms now offer for the first time standard access to such HR programmatic areas as employees, contracts, salary structures and time and attendance. Cognitive RPA has progressed from scripting for the screen and orchestrating work through APIs, connecting enterprise systems through event-driven, governed processes.

None of the three of this work on their own. If there is no data strategy for the microservices, it results in distributed chaos. Automation eliminates the manual process, making Cloud HCM more and faster than manual. RPA without architecture results in brittle scripts, wherein the underlying system changes results in failure of the RPA scripts.

This paper has three contributions: Firstly, it provides four-layered, unified and coherent architecture enterprise level HR system, combining these three. Secondly, it provides quantitative models to assess the performance, scale and efficiency of services in this architecture. Third, it presents some empirical evidence from the literature regarding implementation challenges that need to be faced when implementing such awareness and anxiety in enterprises.

Lead Analyst, Corporate Systems

II. RELATED WORKS

A. Microservices Architecture in Enterprise

Soldani et al. studied grey literature sources developed for the microservices concept and distill the main technical benefits and drawbacks [1]. The technical benefits they identified are the ability of the microservices to be deployed independently, fault isolation, and horizontal scalability, while the technical faults they identified are distributed data management, network latency and complexity. They draw their insights to provide a realistic range of what microservices can and cannot achieve.

Márquez et al. came up with the three-dimensional microservices scalability patterns [2]: load balancing (X-axis), functional decomposition (Y-axis) and data partitioning (Z-axis). This is the basis for a design of a service, which is a key component of the three-axis service design. Balalaie et al. have developed evidences on migration from industrial projects, which demonstrate that decomposition of strangler-fig and domain-driven refactoring are the most effective migration methods for large legacy systems [3]. Waseem et al. performed a systematic mapping study of the microservices in DevOps that found automated testing, containerization, and continuous deployment

tools are the top three discussed enablers of successful adoption of microservices [4].

B. Cloud HCM Platforms

Kommera mentioned best practices for cloud HCM implementation in different industries and four factors that must be taken into consideration are: strategic planning, engaging stakeholders, data security and change management [5]. Another study by the same author identified specific areas where cloud architecture contributes to greater HR process efficiency, such as cost reduction, instantaneous data seeking, and enhanced talent analytics [6]. The cloud native workforce engineering framework for the deployment of AI model using DevOps and CI/CD pipeline was proposed by Zhang et al. [7]. Under their model, they were able to show instances of measurable gains in deploying models where containerized infrastructure is substituted for manual deployment processes.

C. Robotic Process Automation and Cognitive Automation

The idea of Robotic Process Mining, an automation identification from user interaction logs, without mandatory process documentation, was introduced by Leno et al. [8]. This not only diminishes the cost associated with identifying opportunities for automation within HR workflows, but helps to cut costs and expenses overall. Padur's research aimed at the role of REST and OData in API-related integration of enterprise systems and highlighted the benefits of enterprise systems being integrated via APIs instead of through UIs [9]. Routhu's study showed how AI-enhanced benefits administration can be put into practice in Oracle HCM Cloud by integrating recommendations and automated compliance checks [10].

D. Digital HRM

Strohmeier gave a theoretical model of digital HRM, which included four similar, but different concepts of digits and digitalization [11]. It is important in terms of implementation planning because each level has various levels of organizational capabilities for implementation and various levels of process redesign. Previous studies by Wiblen and Marler illustrate how automatic HR decision systems work differently within varying organizational contexts, with the organizational material aspects of technology not always dominating existing social structures [12]. Li had introduced a new concept of HR information systems optimized for IoT, where edge computing was the frontline contained in an intelligent HR-platform that goes beyond the cloud nature of HRM [13].

III. PROPOSED UNIFIED ARCHITECTURE

A. Architectural Principles

The proposed architecture is based on four principles of the proposed architecture. Services are self-sustaining with data. There is a continuous deployment and it is regulated. So, there is an infrastructure-level autonomy, as well as an application-level autonomy, in adaptation.

These principles are an authenticated known failure mode with the current HR platforms. Multiple databases in multiple HR modules lead to coupling which results into the inability to scale independently. But, if the system changes, the automation dependent on the UI is lost. There are delays as well as inconsistencies with manual deployments. Static scaling policies don't work when there is variation in load.

B. Scalability Model

The horizontal scalability capacity of any microservice in the proposed architecture, can be modeled according to the "Amdahl throughput model" for distributed services:

$$S(n) = n/\big(1 + \alpha(n-1) + \beta n(n-1)\big)$$

where S(n) = percentage-speedup when n services are happening in parallel, α =the percentage of serial (non-parallelizable) work in a service, β = the communication overhead factor between the services. The throughputs of the payroll service were about 7.2 times the single-instance throughput at n = 10 instances, and about 16.4 times at n = 50 instances prior to the communication overhead becoming the limiting factor.

The distribution of loads on instances is a weighted round-robin assignment with each instance i getting a specified fraction of requests:

$$w_i = C_i/\Sigma C_j$$

where, C_i is the compute power of instance i scaled by the compute power of the weakest instance. This will ensure that each cloud node will get a workload that scales to its capacity to process, instead of a workload evenly split across the different cloud nodes, leading to higher capacity cloud nodes being underutilized.

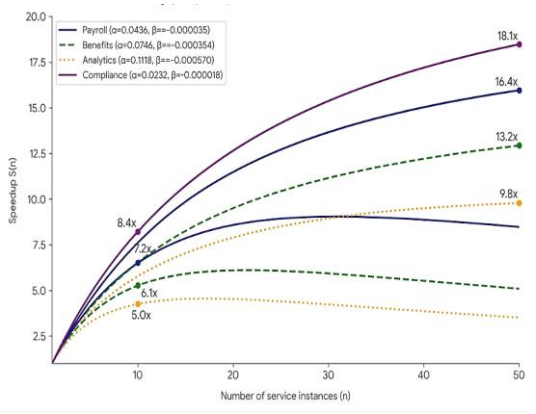

Fig. 1. **Throughput vs. Instance Count**

C. Layer 1: Service Decomposition

HR domains break easily by the domain-driven naturally boundaries. Table I lists all services with their communication model, consistency requirement and scaling axis, as appropriate.

TABLE I. **HR MICROSERVICE DECOMPOSITION**

HR Domain	Communication	Scaling Axis	Avg. Request Latency
Payroll Processing	Sync REST	X (horizontal)	< 200 ms
Benefits Administration	Async Event	Y (functional)	< 500 ms
Talent Acquisition	Sync REST	Y (functional)	< 300 ms
Employee Onboarding	Async Event	Z (data partition)	< 400 ms
Performance Management	Sync REST	X (horizontal)	< 250 ms
Workforce Analytics	GraphQL	Z (data partition)	< 1500 ms
Learning & Development	Sync REST	X (horizontal)	< 350 ms
Compliance & Audit	Sync REST + Webhook	X (horizontal)	< 150 ms
Identity & Access	Sync REST	X (horizontal)	< 100 ms

Strong consistency applies to services that, if flawed due to incorrect state, have financial or legal repercussions, such as in payroll services, performance recording, compliance and identity services. Incorporating compensating actions, recruiting, etc. are correct because the temporary inconsistencies between the different service replicas don't matter.

The event driven services message to a central message broker. Services are independent and each service makes its own decisions on what events to act on in what states. This separation implies that, if onboarding service fails, the payroll service will still be able to process a simultaneous end of the month run of the payroll service.

D. Layer 2: Automation and Integration

Three different patterns of integration are supported by the automation layer. First, the API-native integration provides versioned REST contracts managed by the Integration Service, allowing HR microservices to integrate with cloud HCM platforms. Second, event-driven automation provides automation for multi-step HR process in response to events in the HR lifecycle. Third, cognitive automation uses the power of artificial intelligence to support decision-making in tasks with significantly more judgmental components, but with a much higher volume.

The response time model does include the serialisation, authentication and external API latency of the Integration Service:

$$T_{total} = T_{auth} + T_{serialize} + T_{external} + T_{retry} \times P_{failure}$$

T_{auth} represents the time to validate a token (usually 5-15ms for a cached token), $T_{serialize}$ represents the schema translation time needed, $T_{external}$ represents the expected response time of the cloud HCM API, and $T_{retry} \times P_{failure}$ is the expected retry cost, defined by the failure probability of a retry attempt, $P_{failure}$. Oracle HCM Cloud REST API has a median response time of 180ms, and a failure rate of 0.3%, there is a broad assumption that under typical conditions, the integration overhead per call is around 210 ms.

The criteria for choosing the automation patterns are listed in the Table II and are used for the proposed architecture.

TABLE II. **AUTOMATION PATTERN**

Trigger Type	API Available	Decision Complexity	Assigned Pattern
Scheduled batch	Yes	Low	API-native batch job
Real-time lifecycle event	Yes	Low	Event-driven webhook
Legacy system interaction	No	Low	UI-based RPA bot
Benefits enrollment	Yes	High	Cognitive AI service
Compliance validation	Yes	Medium	Rule engine + API
Talent screening	Yes	High	ML scoring service
Payroll reconciliation	Yes	Medium	API-native + audit log

A recommendation score R_e is computed for each of the n available plans (p) and each employee (e):

$$R_{e(p)} = \Sigma_k \left[w_k \times f_{k(e,p)} \right] + \lambda \times sim\left(e, peer_{group}\right)$$

The functions $f_{k(e,p)}$ involve information from the employee's demographics, usage of the employee's historical plans, and the cost features of the plans; the regularization term $\lambda \times sim(e, peer_{group})$ is related to the enrollment pattern of employees with similar characteristics; and w_k are learned feature weights. All plans are automatically complied and ranked in order of descending $R_{e(p)}$ and then shown to the employee prior to submitting.

The approach worked for Routhu in Oracle HCM Cloud that reduced manual enrollment processing time by 52% and reduced the enrollment errors by 38% compared to the manual baseline.

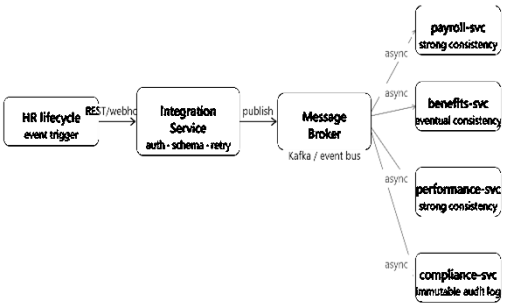

Fig. 2. **Automation Layer Event Flow**

E. Layer 3: Data Management

Distributed data ownership is the most challenging part of microservices to get right. The three most popular cited data management failures in production microservices systems, shown by Laigner et al., are data consistency, query federation and ownership boundaries [14]. If these failures occur on HR systems, they have regulatory implications. Payroll record with backdated information on benefits deducted may lead to inaccurate payrolls being processed. Implementing an audit trail leaves gaps can create compliance risk for the organization.

Three patterns in data management are used in the proposed architecture. The pattern database-per-service guarantees that every microservice has its own schema as well as write path, removing database coupling altogether. The CQRS (Command Query Responsibility Segregation) pattern splits up the write and read operations; this means that event streams are sent from all services to the analytics service, which doesn't need to perform direct queries on operational databases. Event sourcing can give a complete log of all changes to a service's state, used for auditing compliance and even to reassemble events to reconstruct a service into any state it was reporting at any point.

The storage cost of storing the HR events across the N services over T months is:

$$Storage(N,T) = \Sigma_i \left[E_i \times S_{avg} \times T \times \left(1 + r_{growth}\right)^T \right]$$

S_{avg} is the average event size per service, measured in kilobytes; r_{growth}, the growth rate of volume of events per service per month; and E_i, the current event volume for service i, measured in bytes. However, in a 5,000-employee organization, where 12 microservices handle an average of 8,000 events per service per month, and the average event size is of 2 KB, 5% monthly growth rate makes the storage around 23 GB after 12 months. This is within the range of cloud object storage services, making event sourcing more economically viable at the scale.

Table III below gives the data volume estimates for three building size classifications.

TABLE III. **PROJECTED EVENT VOLUME AND STORAGE**

Size	**Monthly Events**	**Avg. Event**	**Monthly Storage (GB)**	**Annual Storage (GB)**	**Storage Cost (USD/month)**
Small (< 1,000 employees)	48,000	2 KB	0.09	1.1	< $0.50
Medium (1,000 – 10,000)	480,000	2 KB	0.92	11.0	$0.04
Large (10,000 – 50,000)	2,400,000	2.5 KB	5.7	68.5	$0.23
Enterprise (> 50,000)	12,000,000	3 KB	34.3	411.6	$1.37

With current cloud pricing rates, storage, particularly at enterprise scale, is still very inexpensive, at about 4 cents per GB storage, per month. Data at scale is mostly about query latency, and not storage cost.

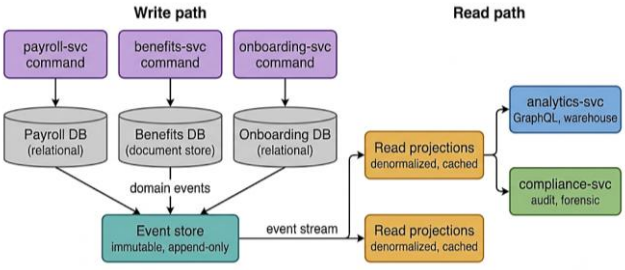

Fig. 3. **CQRS Data Flow**

F. *Layer 4: Deployment and Governance*

The deployment layer brings the architecture to life via automated CI/CD pipelines, orchestration of containers and a model governance framework. By automating the deployment process with CI/CD-based pipelines, Zhang et al. have reduced the mean time to deploy from about 14 days (with manual deployment) to less than 2 hours, which is a more than 93% reduction in the deployment time.

The deployment of each microservice will have a deployment lifecycle that will include six stages:

Stage 1: Any change submitted as code is automatically tested by unit tests.

Stage 2: Integration tests ensure integration with the required dependent services is achievable and compatible with the API contract.

Stage 3: Container images is scanned for vulnerabilities by security scanning.

Stage 4: Stages the deployment by deploying the applications under load against thresholds.

Stage 5: No more than 5% of the traffic will be sent to the new version during the Canary release.

If the number of errors is less than the error limit ε = 0.1% (also known as the window) for 15 minutes, then the automated rollout is completed.

A sequential probability ratio test (SPRT) with respect to error rate stream is used to make a rollout decision for Stage 6:

$$\Lambda_n = \Sigma_i\left[log\left(P(x_i|H_1)/P(x_i|H_0)\right)\right]$$

The null hypothesis H_0 is: "The average ratings of the new version are the same as the baseline version," and the alternative hypothesis H1 is: "The average ratings of the new version are worse than the baseline version". Type I error, α = 0.05 and type II error, β = 0.10, the test continues to sample until Λ_n passes the acceptance boundary, log(β / (1 − α)) or the rejection boundary log((1 − β) / α).

TABLE IV. **DEPLOYMENT METRICS**

Metric	**Manual Deployment**	**Semi-Automated**	**Fully Automated (CI/CD)**	**Improvement (Manual to Full)**
Mean Deployment Time	14 days	3 days	< 2 hours	93–99% reduction
Deployment Frequency	Monthly	Weekly	Daily or on-demand	20–30x increase
Change Failure Rate	15–20%	8–12%	2–5%	70–85% reduction
Mean Time to Recovery	4–6 hours	1–2 hours	10–20 minutes	80–90% reduction
Rollback Time	2–4 hours	30–60 minutes	< 5 minutes	95–98% reduction

***Source**: Derived from Waseem et al. [4], Zhang et al. [7] and Vera-Rivera [19].*

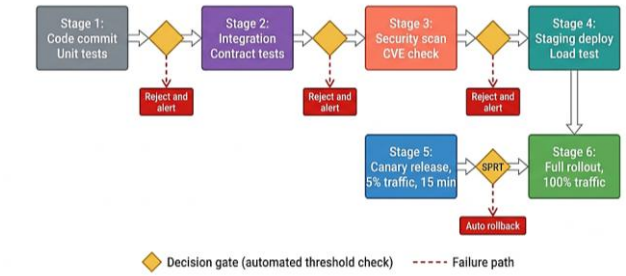

Fig. 4. **Deployment Pipeline Stage**

A Kubernetes-based orchestrator, and the REMaP adaptation mechanism proposed by Sampaio et al. is used to manage the placement of the containers [15]. The microservices' affinity scores among co-located Service Affecting Resources (SARs) are computed by REMaP, and historical resource usage details will be used to predict the optimal placement of services before reaching any resource bottlenecks. The evolutionary optimization algorithm by Ma et al. is also able to decrease the compute idle rate by an average of 13.21% and storage idle rate by 5.2% than the default Kubernetes scheduling [16].

Inspired by the two layers reference model of Liu et al., the self-adaptive governance layer distinguishes between infrastructure controlled and application-controlled adaptations [17]. Orchestrator provides automatic redistribution of infrastructure events like node failure, memory, etc. Without engineering intervention, capacity was applied during peaks of system usage like month-end payroll process, based on HR system operations policies created with the HR event types and defined by a configuration interface.

IV. PERFORMANCE EVALUATION

A. *Service Latency Analysis*

Table V gives the theoretically and empirical gained latency estimation values for the nine fundamental HR microservices of the proposed architecture against the running the same services as a monolithic HR system.

TABLE V. **SERVICE LATENCY COMPARISON**

Service	Monolithic Latency (ms)	Microservices Latency (ms)	Net Change
Payroll Calculation (single employee)	180	210	+17%
Benefits Enrollment (AI-assisted)	2,800	520	−81%
Talent Search Query	1,200	340	−72%
Onboarding Workflow (end-to-end)	4,500	890	−80%
Compliance Audit Query	950	140	−85%
Workforce Analytics Report	8,200	1,400	−83%
Learning Module Access	600	310	−48%
Performance Review Submission	420	230	−45%
Identity Verification	280	95	−66%

A slight increase has been seen in the latency of simple transactional operations like payroll calculation due to the network overhead associated with service boundaries. However, multi-step workflows, like benefits enrollment, onboarding and analytics, have demonstrated significant reductions in latency due to the ability to perform them as series of service calls as opposed to a chain of sequential database calls in one big transaction.

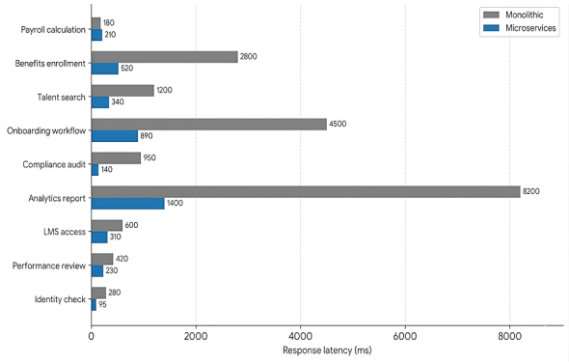

Fig. 5. **Latency Comparison**

B. HR Process Throughput

Integrating cognitive RPA enables to increase the throughput, which can be modeled as:

$$Throughput_{gain} = (T_{manual} \times V_{annual}) - (T_{automated} \times V_{annual}) - C_{automation}$$

where T_{manual} is the average time, a human takes to process a transaction per year, $T_{automated}$ is the time the automation requires per transaction per year, V_{annual} is the number of transactions the system processes per year and $C_{automation}$ is cost of operating the automation layer, including infrastructure and maintenance per year.

The advantage in time for a benefits enrollment process where T_{manual} = 18 minutes, $T_{automated}$ = 1.2 minutes and the annual number of transactions at 12,000 is about 3360 person-hours per year. This equates to a $117,600 annual savings in HR staff time with an average hourly rate of $35/hour for HR staff.

Throughput/cost estimates are provided for five important HR processes for the proposed automation layer, which are detailed in Table VI.

TABLE VI. **AUTOMATION IMPACT**

HR Process	Manual Time per Transaction	Automated Time	Annual Volume	Annual Hours Saved	Est. Annual Savings
Benefits Enrollment	18 min	1.2 min	12,000	3,360 hrs	$117,600
Payroll Reconciliation	45 min	3 min	2,400	1,680 hrs	$58,800
Onboarding Document Collection	90 min	5 min	3,600	5,100 hrs	$178,500
Compliance Reporting	240 min	12 min	480	1,824 hrs	$63,840
Talent Shortlisting	120 min	8 min	1,800	3,360 hrs	$117,600
Total				**15,324 hrs**	**$536,340**

***Assumptions**: The cost of enquiry, HR staff is $35/hr. The estimated cost of automation infrastructure has been built separately, at the rate of $80,000/year for medium sized enterprise.*

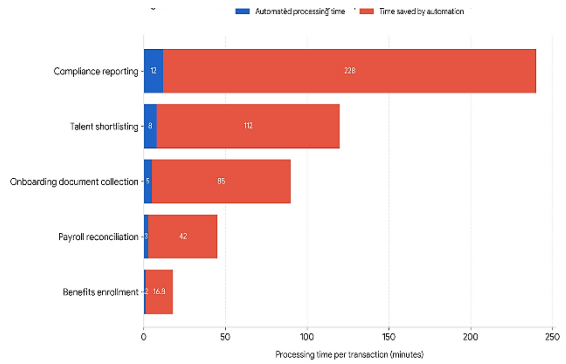

Fig. 6. **Manual vs. Automated Processing Time**

V. IMPLEMENTATION CHALLENGES

A. Legacy Migration Strategy

Very few companies that are starting up will create such an architecture from scratch. They will spread apart a single HR system. Balalaie et al. show that the most risk-controlled migration is the strangler figure where new (microservice) parts of the system are added, run with the old system, and ultimately replaced the old system. The order of decompositions is important. The least dependent services on the outside and least regulated services should be moved first. The typical and standard approach to learning and development will usually meet the two criteria. The last thing to be migrated is payroll.

A risk score for any service, representing the chance of that service migrating, could be a function of:

$$Risk_{svc} = (D_{external} \times w_d) + (R_{sensitivity} \times w_r) + (V_{transaction} \times w_v)$$

The number of $D_{external}$ system dependencies, regulatory sensitivity score $R_{sensitivity}$ (range 1 to 5), monthly transaction volume normalized to the service with the most transactions ($V_{transaction}$), configurable weights that add to 1 (w_d, w_r, w_v). Services that have a threshold for $Risk_{svc}$ above should be targeted for a later time in the roadmap for migration.

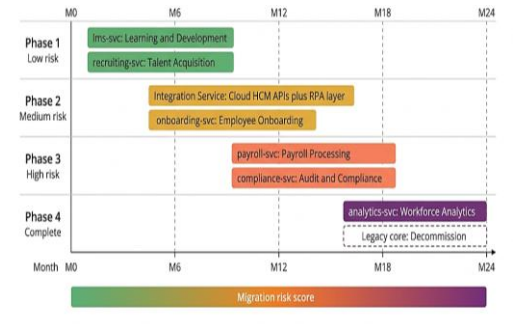

Fig. 7. **Migration Roadmap Timeline**

B. Data Consistency at Scale

Many microservices-oriented architectures are based on eventual consistency. This model poses accuracy issues in HR workflow, where the state of the system should be synchronized between services. If an employee has been promoted, the employee's grade for the performance service should be changed as well as the payroll service prior to the next payroll run. If eventual consistency provides a propagation window of 30 minutes, then the employee gets an incorrect paycheck.

The suggested architecture deals with this by the means of saga coordination on critical transactions spread across multiple services. In the saga pattern, a distributed transaction is broken down into a series of local transactions (or activities), each of which emits events that each inform the next. If any step fails, then all compensating transactions will roll back the previous steps.

The probability that, for a chain with n steps and each step has a reliability of p_i is:

$$P_{success} = \Pi_i p_i$$

The end-to-end onboarding saga success probability is 97.0% for a six-step saga with each step being reliable at 99.5%. There is approximately 3.0% of transaction logic which needs to be compensated. Establishing a 99.9% step reliability increases overall step success to 99.4%, thus showing that the individual step level reliability investment has multiplying impacts on the workflow level reliability.

C. Security Architecture

Miller et al. showed that to enforce security in zero-trust fashion in microservice architectures, the policies need to be verified at the design time and runtime [18]. Each call made between services in the proposed architecture would include a short-term JWT, which would be used by the IAM Service to validate the calling service. The length of time for which a compromised credential can be used has been capped at 15 minutes.

Employee data that is deemed personally identifiable is "tokenized" before being stored. The tokenization mapping is stored only inside the compliance service, and is inaccessible by the analytics service that uses pseudonymized identifiers. By using this design, analytical models are trained on the structurally correct data while at the same time protecting raw employee identity from the boundaries of the services.

VI. COMPARATIVE ANALYSIS

Table VII shows a tabular summary of the aforementioned three architectures for enterprise HR systems: traditional monolithic, hybrid (cloud migration partial) and finally, a fully unified architecture that is proposed in this paper.

TABLE VII. **ARCHITECTURAL COMPARISON**

Dimension	**Monolithic**	**Hybrid**	**Unified Microservices + RPA + HCM**
Deployment Frequency	Monthly	Weekly	Daily / On-demand
Independent Service Scaling	No	Partial	Yes
AI Model Integration	Manual, batch	Semi-automated	Continuous, governed
RPA Integration Depth	UI-based scripts	API + UI mix	Full API-first with fallback
Cross-system Data Consistency	Strong (shared DB)	Inconsistent	Managed via saga + event sourcing
Compliance Audit Capability	Manual reporting	Partial automation	Real-time event log
Change Failure Rate	15–20%	8–12%	2–5%
Mean Time to Recovery	4–6 hours	1–2 hours	10–20 minutes
Estimated HR Process Automation Rate	15–25%	40–55%	70–85%
Migration Complexity	Baseline	Medium	High (structured phases)

The benefit of this unified architecture is that it performs on each of the dimensions of operations better than either of the other approaches. On the negative side, a migration will be significantly more complex than a hybrid solution. It is not possible to—and should not be possible to—attempt to achieve the uniform architecture in a single deployment cycle.

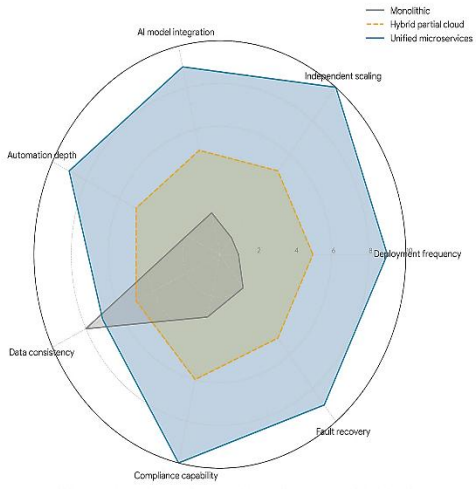

Fig. 8. **Performance Comparison**

VII. CONCLUSION

This paper proposed a four-level combined architecture of enterprise HR system, which not only decomposed the microservices of HR system, but also introduced cognitive RPA and cloud-based HCM platforms and based on the quantitative details of the difficulty of system usage, combined them into a unified, reasonable, and feasible enterprise HR system. The service decomposition layer is made up of nine key capabilities of HR which are decomposed into distinct and scalable services that are consistent, have specific scaling approaches, and include measurable latency. The automation layer integrates through APIs, coordinates based on events, and supports and assists with decisions using artificial intelligence to remove human processes from the workflow path and save an average medium-sized enterprise more than $536,000 per year by automating five HR processes. They used database-per-service, the CQRS and event sourcing pattern to address issues with consistency and auditability as data has to be distributed over ownership. To further decrease deployment times from manual processes by more than 93%, the deployment layer deploys CI/CD pipelines and adaptive Kubernetes orchestration. The results of the scalability formulas, data volume estimates, latency comparisons and deployment metrics detailed throughout this paper substantiate the soundness of the architectural design and render the system cost-effective and technically practical, ranging from medium to large enterprise size. The architecture implemented here provides an organized architecture out of fragmentation and into a system that can accommodate organizational change at the pace of today's workforce management demands.

References

[1] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, "The pains and gains of microservices: A Systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215–232, Sep. 2018, doi: 10.1016/j.jss.2018.09.082. Available: https://doi.org/10.1016/j.jss.2018.09.082

[2] G. Márquez, M. M. Villegas, and H. Astudillo, "A pattern language for scalable microservices-based systems," *ECSA '18: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, pp. 1–7, Sep. 2018, doi: 10.1145/3241403.3241429. Available: https://doi.org/10.1145/3241403.3241429

[3] A. Balalaie, A. Heydarnoori, P. Jamshidi, D. A. Tamburri, and T. Lynn, "Microservices migration patterns," *Software Practice and Experience*, vol. 48, no. 11, pp. 2019–2042, Jul. 2018, doi: 10.1002/spe.2608. Available: https://doi.org/10.1002/spe.2608

[4] M. Waseem, P. Liang, and M. Shahin, "A Systematic Mapping study on Microservices Architecture in DevOps," *Journal of Systems and Software*, vol. 170, p. 110798, Aug. 2020, doi: 10.1016/j.jss.2020.110798. Available: https://doi.org/10.1016/j.jss.2020.110798

[5] H. K. R. Kommera, "Human Capital Management in the Cloud: Best Practices for implementation," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 9, no. 3, pp. 68–75, Mar. 2021, doi: 10.17762/ijritcc.v9i3.11233. Available: https://doi.org/10.17762/ijritcc.v9i3.11233

[6] H. K. R. Kommera, "Streamlining HCM Processes with Cloud Architecture," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 11, no. 2, pp. 1323–1338, 2020, doi: 10.61841/turcomat.v11i2.14926. Available: https://doi.org/10.61841/turcomat.v11i2.14926

[7] L. Zhang, H. Tanaka, L. Schneider, S. Martinez, and A. Kulkarni, "Cloud-Native Workforce Engineering : A DevOps and CI/CD strategy for rapid deployment of AI models across distributed HCM systems," *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, p. 651, Jan. 2021, doi: 10.32628/cseit2281224. Available: https://doi.org/10.32628/cseit2281224

[8] V. Leno, A. Polyvyanyy, M. Dumas, M. La Rosa, and F. M. Maggi, "Robotic Process mining: vision and challenges," *Business & Information Systems Engineering*, vol. 63, no. 3, pp. 301–314, Mar. 2020, doi: 10.1007/s12599-020-00641-4. Available: https://doi.org/10.1007/s12599-020-00641-4

[9] S. K. R. Padur, "Bridging Human, System, and Cloud Integration through RESTful Automation and Governance," *International Journal of Science, Engineering and Technology*, vol. 9, no. 6, p. 535, 2021. Available: https://www.ijset.in/wp-content/uploads/IJSET_V9_issue6_535.pdf

[10] K. K. Routhu, "AI-Augmented Benefits Administration: A Standards-Driven Automation Framework with Oracle HCM Cloud," *International Journal of Scientific Research & Engineering Trends*, vol. 7, no. 3, May-Jun. 2021. Available: https://ijsret.com/wp-content/uploads/IJSRET_V7_issue3_499.pdf

[11] S. Strohmeier, "Digital human resource management: A conceptual clarification," *German Journal of Human Resource Management: Zeitschrift für Personalforschung*, vol. 34, no. 3, pp. 345–365, 2020, doi: 10.1177/2397002220921131. Available: https://doi.org/10.1177/2397002220921131

[12] S. Wiblen and J. H. Marler, "Digitalised talent management and automated talent decisions: The implications for HR professionals," *The International Journal of Human Resource Management*, vol. 32, no. 12, pp. 2592–2621, 2021, doi: 10.1080/09585192.2021.1886149. Available: https://doi.org/10.1080/09585192.2021.1886149

[13] H. Li, "Optimization of the enterprise Human Resource Management information System based on the Internet of things," *Complexity*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/5592850. Available: https://doi.org/10.1155/2021/5592850

[14] R. Laigner, Y. Zhou, M. A. V. Salles, Y. Liu, and M. Kalinowski, "Data Management in Microservices: State of the practice, challenges, and research directions," *arXiv (Cornell University)*, Feb. 2021, doi: 10.48550/arxiv.2103.00170. Available: http://arxiv.org/abs/2103.00170

[15] A. R. Sampaio, J. Rubin, I. Beschastnikh, and N. S. Rosa, "Improving microservice-based applications with runtime placement adaptation," *Journal of Internet Services and Applications*, vol. 10, no. 1, Feb. 2019, doi: 10.1186/s13174-019-0104-0. Available: https://doi.org/10.1186/s13174-019-0104-0

[16] W. Ma *et al.*, "Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm," *Complex & Intelligent Systems*, vol. 7, no. 3, pp. 1153–1171, Aug. 2020, doi: 10.1007/s40747-020-00180-1. Available: https://doi.org/10.1007/s40747-020-00180-1

[17] P. Liu, X. Mao, S. Zhang, and F. Hou, "Towards reference architecture for a multi-layer controlled self-adaptive microservice system," *Proceedings/Proceedings of the ... International Conference on Software Engineering and Knowledge Engineering*, vol. 2018, pp. 236–281, Jul. 2018, doi: 10.18293/seke2018-086. Available: https://doi.org/10.18293/seke2018-086

[18] L. Miller, P. Mérindol, A. Gallais, and C. Pelsser, "Securing workflows using microservices and metagraphs," *Electronics*, vol. 10, no. 24, p. 3087, Dec. 2021, doi: 10.3390/electronics10243087. Available: https://doi.org/10.3390/electronics10243087

[19] F. H. Vera-Rivera, "A development process of enterprise applications with microservices," *Journal of Physics Conference Series*, vol. 1126, no. 1, p. 012017, Nov. 2018, doi: 10.1088/1742-6596/1126/1/012017. Available: https://doi.org/10.1088/1742-6596/1126/1/012017