DTGov: Digital Transformation of Government Business Processes

Nuno Marques and André Vasconcelos
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Keywords: Digital Transformation, Government, Processes, Low-Code, Reference Architecture

Abstract: Today, companies cannot rely on outdated systems, governments being no exception. There are multiple benefits when performing the digital transformation (DT) of its processes, but also several downsides such as costs. This paper presents first the main use cases: resolve a procedure, send a notification by hand, reclassify a procedure, and create a request, and then the reference solution divided into 3 components. First, the Domain Model and Lifecycles related to the main entities. Second, the Reference Architecture adds the remaining projects' modules (Roles,...), and also the dependencies between them. Third, the Implementation presents how to implement the information in a real-world case using "low-code" technology, especially useful to mitigate several issues. Lastly, an evaluation of the proposed solution is also presented.

1 INTRODUCTION

Digital Transformation is crucial in today's world, as seen in the biggest companies like Amazon (Kimberling, E., 2021), and governments are no exception.

One possible way to diminish many of the existing problems is through the usage of low-code technologies, since they focus on the feedback approach, and simpler development methods. One issue, however, is that since this technology is relatively new, information regarding its implementation in real-world scenarios is scarce.

With that in mind, the goal here is to explain how to implement the DT of government processes using Enterprise Information Management (EIM) systems and low-code solutions. The work presented here focuses on the employees' application and is based on a real-world implementation using OpenText AppWorks (OpenText, nda).

This paper first presents the related work containing a Systematic Literature Review (SLR) that identifies the principal modules used in similar DTs. From that, and the real-world implementation mentioned, the main use cases are identified. Following that, it is created the solution proposition, divided into 3 parts: entity models and their lifecycles, reference architecture, and implementation using low-code technologies. Lastly, there is the work's evaluation and conclusion, mentioning this paper's main contributions and possible improvements in future work.

2 RELATED WORK

2.1 Background

Before the presentation and explanation of the SLR performed, some main topics, mentioned in this paper, need to be acknowledged.

2.1.1 (Business) Process Management

Business Process Management "is essential to understand the steps that are ... part of a business process, ... the people who are involved in these steps, the information that is being exchanged and processed ..., and the technologies that are invoked when executing the various steps." (Reijers, H.A., 2021) In other words, it uses activities, tasks, and others, towards a specific goal, for example, the payment of a procedure. Additionally, it also comprises the management, technologies, and people related to the process.

2.1.2 Low-Code Approaches

Low-code, as mentioned before, is a "visual approach to software development, " (mendix, nd). This means that programmers do not need to be highly specialized, which can be hard to find and costly, and the development of the application is usually faster, more interactive, and more agile. Due to the visual aspect, stakeholders can better understand what is being created and propose changes accordingly.

Marques, N. and Vasconcelos, A.
DTGov: Digital Transformation of Government Processes.
DOI: 10.5220/0011715700003467
In *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS 2023) - Volume 2*, pages 101-108
ISBN: 978-989-758-648-4; ISSN: 2184-4992

2.2 SLR: Modules Present in the DT of Government Processes

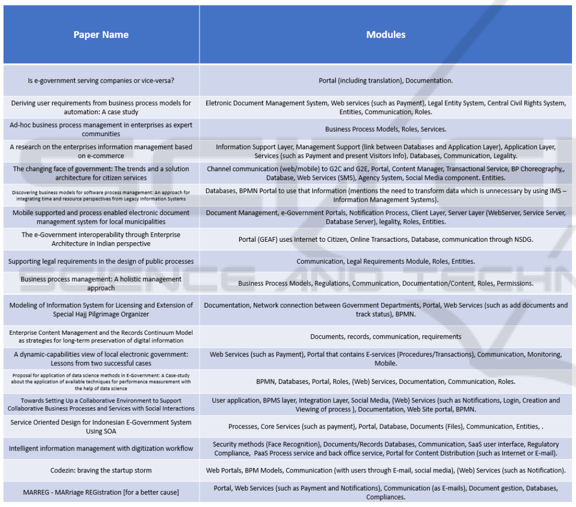
This SLR aims to respond to the question, what are the main modules, used by Enterprise Information Management Systems, in the Digital Transformation of government's procedures/processes/functions?

The steps performed were search process (query "((Enterprise AND Information AND Management) AND (System OR Solutions)) AND Government AND ((Digital AND Transformation) OR (Process OR Automation) OR Digitalization OR Low-Code OR (Low AND Code))"), filtering process, data extraction and quality assessment.

State of the Art

From the SLR performed, the information obtained (and presented in Table 1) is used to create the following state-of-the-art:

Table 1: Modules present in each paper from the second SLR.



Paper Name	Modules
Is e-government serving companies or vice-versa?	Portal (including translation), Documentation.
Deriving user requirements from business process models for automation: A case study	Eletronic Document Management System, Web services (such as Payment), Legal Entity System, Central Civil Rights System, Entities, Communication, Roles.
Ad-hoc business process management in enterprises as expert communities	Business Process Models, Roles, Services.
A research on the enterprises information management based on e-commerce	Information Support Layer, Management Support (link between Databases and Application Layer), Application Layer, Services (such as Payment and present Visitors Info), Databases, Communication, Legality.
The changing face of government: The trends and a solution architecture for citizen services	Channel communication (web/mobile) to G2C and G2E, Portal, Content Manager, Transactional Service, BP Choreography,, Database, Web Services (SMS), Agency System, Social Media component. Entities.
Discovering business models for software process management: An approach for integrating time and resource perspectives from Legacy Information Systems	Databases, BPMN Portal to use that Information (mentions the need to transform data which is unnecessary by using IMS – Information Management Systems).
Mobile supported and process enabled electronic document management system for local municipalities	Document Management, e-Government Portals, Notification Process, Client Layer, Server Layer (WebServer, Service Server, Database Server), legality, Roles, Entities.
The e-Government interoperability through Enterprise Architecture in Indian perspective	Portal (GEAF) uses Internet to Citizen, Online Transactions, Database, communication through NSDG.
Supporting legal requirements in the design of public processes	Communication, Legal Requirements Module, Roles, Entities.
Business process management: A holistic management approach	Business Process Models, Regulations, Communication, Documentation/Content, Roles, Permissions.
Modeling of Information System for Licensing and Extension of Special Hajj Pilgrimage Organizer	Documentation, Network connection between Government Departments, Portal, Web Services (such as add documents and track status), BPMN.
Enterprise Content Management and the Records Continuum Model as strategies for long-term preservation of digital information	Documents, records, communication, requirements
A dynamic-capabilities view of local electronic government: Lessons from two successful cases	Web Services (such as Payment), Portal that contains E-services (Procedures/Transactions), Communication, Monitoring, Mobile.
Proposal for application of data science methods in E-Government: A Case-study about the application of available techniques for performance measurement with the help of data science	BPMN, Databases, Portal, Roles, (Web) Services, Documentation, Communication, Roles.
Towards Setting Up a Collaborative Environment to Support Collaborative Business Processes and Services with Social Interactions	User application, BPMS layer, Integration Layer, Social Media, (Web) Services (such as Notifications, Login, Creation and Viewing of process), Documentation, Web Site portal, BPMN.
Service Oriented Design for Indonesian E-Government System Using SOA	Processes, Core Services (such as payment), Portal, Database, Documents (Files), Communication, Entities, .
Intelligent information management with digitization workflow	Security methods (Face Recognition), Documents/Records Databases, Communication, SaaS user interface, Regulatory Compliance, PaaS Process service and back office service, Portal for Content Distribution (such as Internet or E-mail).
Codezin: braving the startup storm	Web Portals, BPM Models, Communication (with users through E-mail, social media), (Web) Services (such as Notification).
MARREG - MARriage REGistration [for a better cause]	Portal, Web Services (such as Payment and Notifications), Communication (as E-mails), Document gestion, Databases, Compliances.

1. Each web page should have different security/permissions, (Aysolmaz, B. and Demirörs, O., 2014), to restrict each employee to their specific task/role. The majority of the papers contain the portal/user page, such as (Mazumder, A., 2020) that mentions an " eGovernance application software that is accessible through a portal by citizens" or (Paul, A. and Paul, V., 2012) that explains that the stakeholders interact with the National portal through the internet and that portal connects to the service providers. Another common trait is that most of them do not mention employee's application.
2. Some entities contain a lifecycle, and documentation (which can also be stored independently). (Aysolmaz, B. and Demirörs, O., 2014) mentions the necessity of "automatically generate user requirements documents". This automation is important and should be incorporated into the process entity's lifecycle.
3. There are also other components, such as web components, that should be stored in a different directory. Examples of communication technologies are web services, which are vital in business process management, and low-coding approaches, such as when sending emails or completing payment, as cited by (Mazumder, A., 2020) that mentions a payment service done "through e-payment receipts portal".

A brief comparison between EIM systems is also performed and from it, it is understood that most low-code solutions use the same components, (leaving up to each government which to choose).

3 USE CASES

Each government is different, having different ways to resolve and deal with its processes. Because of this, and since there are multiple use cases possible, this chapter only presents the ones deemed most relevant, given the SLR and the original real-world implementation.

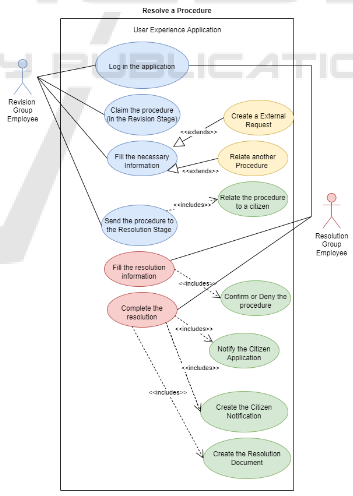

Resolve a Procedure
User Experience Application
Revision Group Employee
Resolution Group Employee
Log in the application
Claim the procedure (in the Revision Stage)
Create a External Request
<<extends>>
Fill the necessary Information
Relate another Procedure
<<extends>>
Relate the procedure to a citizen
<<includes>>
Send the procedure to the Resolution Stage
Fill the resolution information
<<includes>>
Confirm or Deny the procedure
Complete the resolution
<<includes>>
Notify the Citizen Application
<<includes>>
<<includes>>
Create the Citizen Notification
Create the Resolution Document

Figure 1: Use Case Diagram 1: Resolve a Procedure.

The use case Resolve a Procedure, Figure 1, is the main one, appearing the most in the papers identified from the SLR performed, such as (Mazumder, A., 2020), which mentions it when performing and resolving the marriage certificate. The goal is to resolve a procedure. First, the employee (Revision Group) needs to fill in the necessary information and send it to another employee (Resolution Group). This employee will then decide if the resolution is positive or negative, finalizing with various automatic activities.

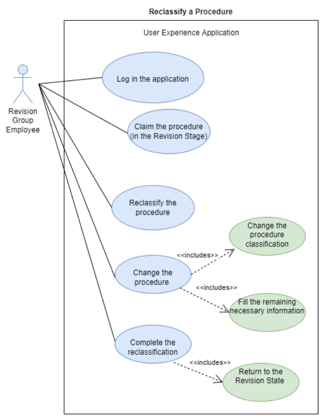

Reclassify a Procedure
User Experience Application
Revision Group Employee
Log in the application
Claim the procedure (in the Revision Stage)
Reclassify the procedure
Change the procedure
Change the procedure classification
Fill the remaining necessary information
Complete the reclassification
Return to the Revision State
<<includes>>
<<includes>>
<<includes>>

Figure 2: Use Case Diagram 2: Reclassify a Procedure.

The other use case is Reclassify a Procedure, Figure 2 which is when a procedure is not created or classified correctly. In this case, the Revision Group Employee sends the procedure to the Reclassify state, where it will be changed and sent back to the Revision state.

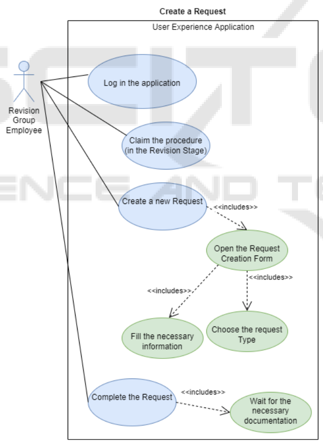


Figure 3: Use Case Diagram 3: Create a Request.

Create a Request, Figure 3, is the use case where more information is necessary, for example from the citizen or an employee. In this case, the Revision Group Employee creates a new request, opening the Creation Request Form and choosing the type of request (internal, external, or citizen), filling in the necessary information. Upon receiving it (such as documentation), the request is completed by the employee.

The last use case identified is the Send a Notification By Hand, Figure 4, which is when the notification needs to be sent through mail or a carrier instead of digitally. After the procedure's resolution is completed, a notification is created automatically.

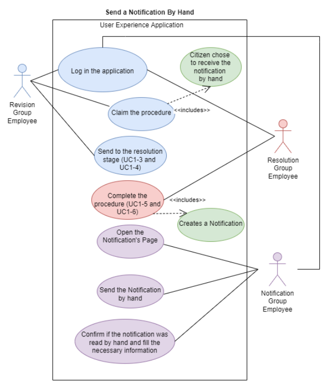

Send a Notification By Hand
User Experience Application
Revision Group Employee
Resolution Group Employee
Notification Group Employee
Log in the application
Citizen chose to receive the notification by hand
Claim the procedure
Send to the resolution stage (UC1-3 and UC1-4)
Complete the procedure (UC1-5 and UC1-6)
Creates a Notification
Open the Notification's Page
Send the Notification by hand
Confirm if the notification was read by hand and fill the necessary information
<<includes>>
<<includes>>

Figure 4: Use Case Diagram 4: Send a Notification By Hand.

If the citizen chooses to receive that notification by hand, the Notification Group Employee will claim it and proceed to its sending, by filling in the necessary information. Upon receiving the confirmation, the Notification Group Employee fills in the remaining information and completes the notification.

4 SOLUTION PROPOSITION

This chapter presents the solution proposed in this paper, which is divided into 3 parts: Domain Model and Lifecycles, Reference Architecture, and Application Implementation.

4.1 Domain Model and Lifecycles

This section is divided into two, the domain model, explaining the main entities and their relationships, and the entities' lifecycles, detailing the different states and transitions.

4.1.1 Domain Model

The Domain model, Figure 5, contains four main projects which are Common Parts, Procedures, Notifications, and External Entities.

They are named projects because it is the name given by the low-code application used, AppWorks, to designate the different parts (directories).

The Procedures Project contains the entity Procedure Parent. This is the main entity, which is abstract, and refers to the government procedures. It contains the main properties such as procedure ID, type, and the different departments.

The Procedure Child inherits the Procedure Parent and adds a few new properties such as procedure child ID and type.

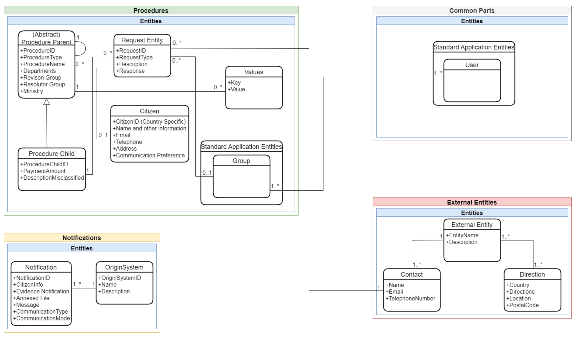


Figure 5: Domain Model.

To simplify future mentions of Procedure Child and Parent, they will be referred as one, Procedure.

There is also the External Entities project, which contains the information relative to each Request from the Procedures entity. External Entities contain the entities External Entity, Contact, and Direction.

Both the Common Parts project and Procedures contain Standard Application Entities, which are entities not created by the developers but by the application, internally.

In the Notifications project, there are two entities, Notification, and OriginSystem. OriginSystem represents the system from which the notification was initially created. The Notification entity refers to the notifications created.

4.1.2 Entities' Lifecycles

Some entities can contain a lifecycle, which comprises their different states and transactions, especially if they have tasks that need to be completed, either manually or automatically. Regarding the ones mentioned, only Procedures (Procedure Child), Figure 6, Request Entity, Figure 7, and Notification, Figure 8, contain them, which will now be mentioned.

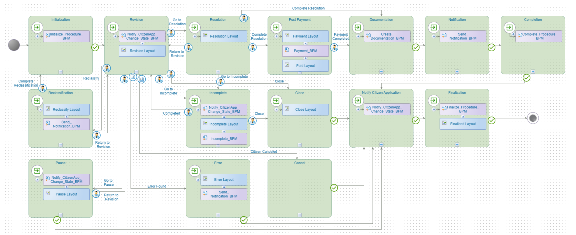


Figure 6: Procedure's Lifecycle Model.

The Procedure starts in the Initialization state, which is where the process to initialize a procedure is done. It comprises the retrieval of the procedure ID from the citizen's application, the payment confirmation, and others.

From this state the procedure goes to the Revision, where the Revision Group employee verifies the information referent to the procedure, adding new information if necessary. From this state, the procedure can go to various paths, including the main one, Resolution, and the alternatives, such as Reclassification.

In the Resolution State, the employee from the Resolution Group decides the procedure's resolution.

From this state, the procedure can go to the Post Payment one (if further payment is required), the Documentation state (where the documentation related to the procedure is created), or the Incomplete state (Resolution Group employee identifies missing information, sending it back).

In the Notification state, a notification is created and sent to the citizen. In the Completion State, the procedure is completed (in the Citizens' application), notifying that the procedure was completed correctly or incorrectly, and performing the last property updates such as date completion.

The last two states, common to almost all states are the Notify Citizen Application state, which is where the citizen is notified of the (previous) state (for example, if an error occurred which cannot be resolved, the citizen is notified of that) and Finalization where the procedure in finalized in the Employees' application, removing the ability to modify the procedure and its documentation (converting all its permissions to read-only).

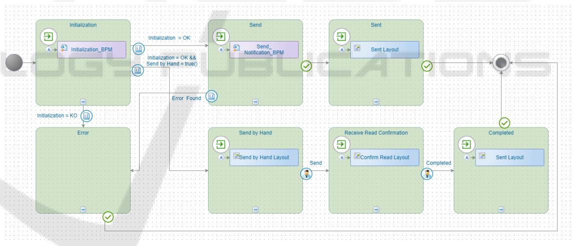


Figure 7: Notification's Lifecycle Model.

The Notifications entity starts similarly to the Procedures (Initialization) but branches into 2: or it can go to the Send state, where the notification is prepared to be sent to the citizen or the Send by Hand if the communication preference chosen is to receive notifications by hand.

From the Send state, it goes to the Sent, where the notification is confirmed correctly sent and its lifecycle finished.

In the Send by Hand, the Notification Group employee revises and confirms the sending of the notification (use case 4). From there, it goes to Receive Read Confirmation, where upon receiving the confirmation that the notification was read, the employee inputs the final values such as read date and confirms its completion. The following and last state is the Completed state, where the notification is confirmed read, and completed, finishing the lifecycle.

There is also another state, achieved from the Initialization and Send, which is the Error state, signaling that some error occurred and aborting the sending of the notification.

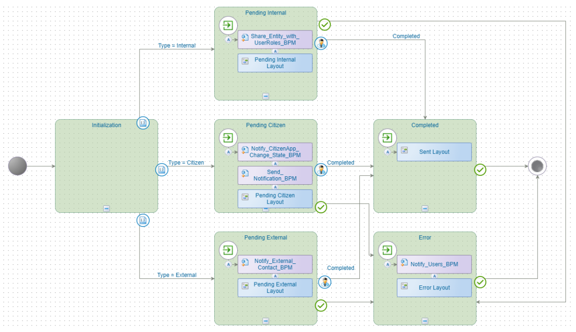


Figure 8: Request's Lifecycle Model.

The Request's lifecycle also starts in the Initialization state, where the identification of its type is done. If Internal, it goes to the Pending Internal state, which communicates with the employee or chosen group. If External, it goes to the Pending External, where is done the communication with the external entity. Finally, if Citizen, it goes to the Pending Citizen, which will communicate with the citizen related to the procedure.

If no error occurred, the request, from all the Pending states, goes to the Completed state, which represents the request sent, finishing the lifecycle. If not, the request goes to the Error state, finishing its lifecycle and not performing the request.

4.2 Reference Architecture

This reference architecture contains, Figure 9, five projects, the four already mentioned and the (External) Citizen Application

4.2.1 Common Parts

The first project, Common Parts, was created with the goal of having all the components that are common to the various projects in a single location, in a way that eases their usage and avoids circular dependencies, (Russo, M. and Ferrari, A., nd). This project contains the module/directory Entities, presented in the previous chapter. It also contains the Business Process Models (BPMs), which are the models used to implement the government's business processes.

The BPMs use the Web Services module, which contains 4 types of web services, internal (created by the application), external (from other applications),

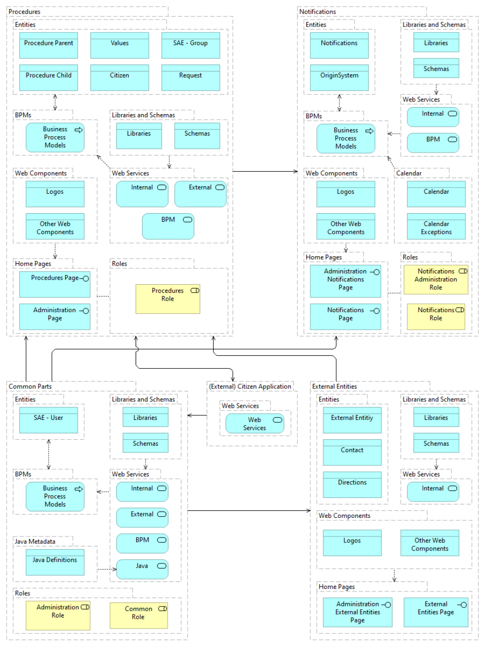


Figure 9: Reference Architecture.

java (from the Java Metadata module), and BPM (from the BPMs created).

Web Services use the Libraries ("collections of prewritten code that users can use to optimize tasks", (Ryan, 2020)) and Schemas modules ("a formal description of the structure or organization of a particular database", (Dearmer, A., 2021)). These can be internal if predefined by the application or external.

Another module present is the Java Metadata, which contains the java definitions. Every program/service that is created externally in Java (or another language if that is the case) is stored here. Normally, these web services are created only if they do not exist in the application used.

Finally, there is the Roles module, containing the roles created, and grouping the employees. In this project, Common Parts, there are two, the Administration Role which receives more permissions, and Common Role which contains more generalized ones.

The following projects will not be explained in so much detail, only being presented the differences and important points.

4.2.2 Procedures

Regarding the Procedures project, they have the same modules as the Common Parts but now the Web Services module does not contain the java BPMs (since they are in the Common Parts). This project also presents a new role, Procedures Role, which gives permissions related to procedures (read, write...).

This project also contains two modules that were not present in the previous one, which are the Web Components and Home Pages.

Home pages are the front end of the application, where the employees can create and resolve procedures. It contains two home pages, Administration Page, reserved only for administrative tasks and the Administrative Role, and the Procedures Page, accessed by the Procedures role users. These home pages can use web components which are, for example, images related to the government.

4.2.3 Other Projects

The Notifications project contains a new role, the Notifications role which is responsible for the notifications, and two new home pages, the Administration Notifications Page and Notifications Page (which is where the tasks to complete the notification's lifecycle are done). Notifications also do not contain external web services because they are already created in the Common Parts.

External Entities project contains a new home page, Administration and External Entities pages, and does not contain Roles or BPMs (especially because it does not have an entity with a lifecycle). The only web services present is internal.

Lastly, there is the (External) Citizen Application, which is a project not created in the employees' application but refereed in this reference architecture to represent the citizens' application. Here, only the web services are relevant, being used in the other projects.

4.3 Application Implementation

Before explaining how to implement the information presented using low-code technologies (in this case AppWorks), first, it is necessary to mention two other EIM solutions, which are Active Directory (regarding OpenText is called Directory Services, OTDS, and Content Management (in this case Content Server, OTCS, (OpenText, nda)).

OTDS allows users, in this case the employees, to have a single sign-on on all the OpenText Applications. More importantly, it contains the possibility to group the users and transfer that information to AppWorks, relating it to the roles module.

OTCS refers to content management. Every document created is stored in this application, which is connected to AppWorks.

Since the majority of the EIM solutions contain both these applications, what is mentioned here can be applied to them.

Regarding AppWorks, to create the various projects, first, it is necessary to configure and create an organization which contains various modules such as entities. Inside an organization, it is possible to create a Collaborative Workspace, which is where ”all modeling activities are done” (OpenText, ndb), the development platform.

Following that, it is necessary to create the different modules, which is done through a file system where each sub-folder, from the project, corresponds to each module mentioned.

The folder Entities contains all the entities present in the Project, with each one containing various building blocks.

Explaining a few of them, web services include the ones related to the entities, BPMs, or other components/modules, and business workspace enables the storage of associated documentation.

The transformation from the Domain Model to the AppWorks is direct, only requiring the creation of the Entity and the necessary building blocks.

Regarding the BPMs module, the information provided does not go in-depth because they are usually very specific to the Government in question.

From the Web Services module, the four types and the information on how to create them (except for the Internal ones which are created through the Entity) were already provided.

Other modules present are the Home Pages and Web Components. AppWorks is divided into the AppWorks Platform, which is where the projects, external web components such as Maps APIs, and others are. This is the developing/administration part (the back end). There is also another part called AppWorks Experience (the front-end), that corresponds to the interface that employees interact.

The last module is Roles. Each user can be assigned one or more roles in the application through a specific component called ”User Manager”. This component contains the roles standard in the AppWorks (administrator, user...) but can also contain the ones created inside the projects. These can have certain permissions, which can restrict the actions that the user performs (such as access to the entity).

The idea here is also to show that low-code technologies allow the development of applications faster and easier than using ”traditional software”, which can help mitigate the various concerns that governments have regarding DT.

5 EVALUATION

Since the development of the software was done in a anonymous government context (which comprised the testing of the application), the evaluation had to be limited. Given this and other limitations such as the topics mentioned being niche, there are not many participants that can evaluate what is proposed.

Mentioning that, it was divided into 3 parts.

5.1 Experts/Juniors Evaluation

This first part was performed by doing an interview with five participants (three experts in the field of DT and usage of low-code, and two juniors), where they answered five open questions, present in the table 2:

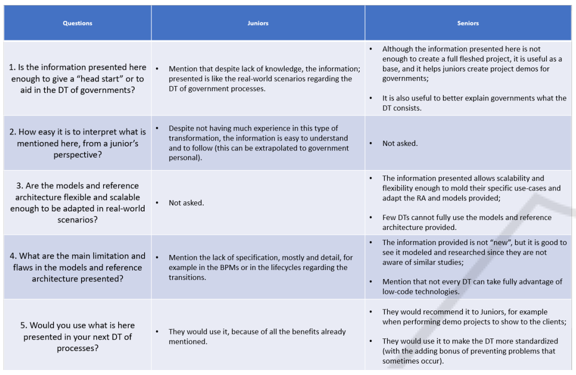


Table 2: Experts/Juniors: Summary of the Findings.

Questions	Juniors	Seniors
1. Is the information presented here enough to give a "head start" or to aid in the DT of governments?	• Mention that despite lack of knowledge, the information; presented is like the real-world scenarios regarding the DT of government processes.	• Although the information presented here is not enough to create a full fleshed project, it is useful as a base, and it helps juniors create project demos for governments; • It is also useful to better explain governments what the DT consists.
2. How easy it is to interpret what is mentioned here, from a junior's perspective?	• Despite not having much experience in this type of transformation, the information is easy to understand and to follow (this can be extrapolated to government personal).	• Not asked.
3. Are the models and reference architecture flexible and scalable enough to be adapted in real-world scenarios?	• Not asked.	• The information presented allows scalability and flexibility enough to mold their specific use-cases and adapt the RA and models provided; • Few DTs cannot fully use the models and reference architecture provided.
4. What are the main limitation and flaws in the models and reference architecture presented?	• Mention the lack of specification, mostly and detail, for example in the BPMs or in the lifecycles regarding the transitions.	• The information provided is not "new", but it is good to see it modeled and researched since they are not aware of similar studies; • Mention that not every DT can take fully advantage of low-code technologies.
5. Would you use what is here presented in your next DT of processes?	• They would use it, because of all the benefits already mentioned.	• They would recommend it to Juniors, for example when performing demo projects to show to the clients; • They would use it to make the DT more standardized (with the adding bonus of preventing problems that sometimes occur).

In summary, experts mention that the information presented allows scalability and flexibility enough to mold specific use cases. Although the information presented is not enough, it is useful to prevent various problems, help juniors create project demos for governments, and to better explain the government personnel what are the goals of the project.

5.2 Developers Evaluation

This evaluation focuses on the developers (four participants, two experts, and two juniors) responsible for the creation of the application, that similarly to the participants from the previous sub-section, answered five open questions in an interview (table 3):

Summarizing, developers report that the major problems were related to limitations, such as bugs. Since the development was faster and could be done in real-time with an interface, clients were satisfied because they felt included and could provide direct feedback. This, in turn, lead to more changes over time. Both groups agreed that using low-code software was very useful and they would choose to use it again if given the choice, because it was faster, easier and despite the learning curve, allowed juniors to work semi-autonomous and without major flaws/errors. They note, however, that the "traditional ways" allow a better understanding of the pro-

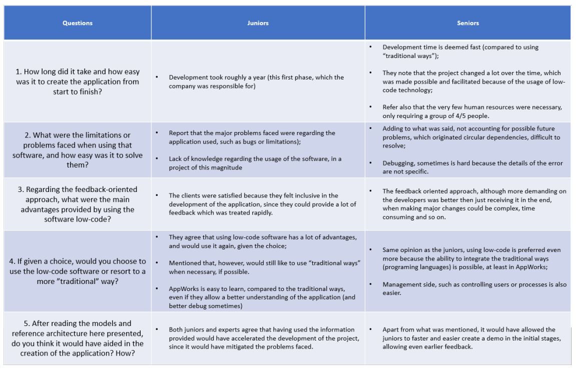


Table 3: Developers: Summary of the Findings.

Questions	Juniors	Seniors
1. How long did it take and how easy was it to create the application from start to finish?	• Development took roughly a year (this first phase, which the company was responsible for)	• Development time is deemed fast (compared to using "traditional ways"); • They note that the project changed a lot over the time, which was made possible and facilitated because of the usage of low-code technology; • Refer also that the very few human resources were necessary, only requiring a group of 4/5 people.
2. What were the limitations or problems faced when using that software, and how easy was it to solve them?	• Report that the major problems faced were regarding the application used, such as bugs or limitations); • Lack of knowledge regarding the usage of the software, in a project of this magnitude	• Adding to what was said, not accounting for possible future problems, which originated circular dependencies, difficult to resolve; • Debugging, sometimes is hard because the details of the error are not specific.
3. Regarding the feedback-oriented approach, what were the main advantages provided by using the software low-code?	• The clients were satisfied because they felt inclusive in the development of the application, since they could provide a lot of feedback which was treated rapidly.	• The feedback oriented approach, although more demanding on the developers was better then just receiving it in the end, when making major changes could be complex, time consuming and so on.
4. If given a choice, would you choose to use the low-code software or resort to a more "traditional" way?	• They agree that using low-code software has a lot of advantages, and would use it again, given the choice; • Mentioned that, however, would still like to use "traditional ways" when necessary, if possible. • AppWorks is easy to learn, compared to the traditional ways, even if they allow a better understanding of the application (and better debug sometimes)	• Same opinion as the juniors, using low-code is preferred even more because the ability to integrate the traditional ways (programing languages) is possible, at least in AppWorks; • Management side, such as controlling users or processes is also easier.
5. After reading the models and reference architecture here presented, do you think it would have aided in the creation of the application? How?	• Both juniors and experts agree that having used the information provided would have accelerated the development of the project, since it would have mitigated the problems faced.	• Apart from what was mentioned, it would have allowed the juniors to faster and easier create a demo in the initial stages, allowing even earlier feedback.

cesses behind the development and, sometimes, a better identification of the existing errors. A balance between using low-code technologies and "traditional methods" would be ideal.

5.3 Related Work Assessment

In this final sub-section, the goal is to compare this paper to others (from the related work) that mention similar DT, and, as mentioned, understand if they could take advantage of the solution proposed here.

The only two papers chosen were (Mazumder, A., 2020) and (Liu, C. and jun tao, X. and Lu, B., 2016) because they are the most similar to this one, containing a DT of processes and development of a real-world application. The idea was to:

1. Compare the theory and development to the other papers and extract the relevant information.
2. Understand if this paper could have benefited them.
3. Understand if low-code software could have fixed or mitigated issues that occurred.

The goal is to understand if this paper is useful to other scholarly papers and real-world scenarios, and if not, identify those flaws and correct them in future work. There are a few important parts that can be retrieved from the papers, such as from (Liu, C. and jun tao, X. and Lu, B., 2016) which mentions a B/S architecture that comprises various tiers or (Mazumder, A., 2020) that mentions that their application follows specific practices and standards.

Regarding this paper's contributions to those papers, they would have allowed not only a faster and simpler development but also improved the feedback approach. Several concerns, such as data encryption or network security, could all be dealt with had the authors followed the information presented here.

6 CONCLUSION

6.1 Main Contributions

In conclusion, from the results obtained, this paper provides valuable information regarding the DT of government processes, by providing a ”template” that can be used by developers to ease and speed the development phase. It can also help clients visualize the work done, allowing a better understanding of the project scope and earlier feedback. The mention of low-code technologies also can provide mitigation of issues that can come from ”traditional methods” such as high complexity or time.

However, since this project was based in a government that wants to remain anonymous and the topics mentioned are niche in the field of IT, a full statistical quantitative evaluation was not possible. Nevertheless, it was realized as a qualitative one, which comprised a few interviews with experts and juniors in the field of DT. Juniors are especially useful because their answers can be extrapolated to the government personnel, which also lacks technical knowledge. From the results, there were major takeaways. First, the models and reference architecture presented are easy to understand and follow from an expert perspective but also a junior/low knowledge one. Second, the information here, despite not presenting new knowledge to the experts in the area, allows them to confirm their development, systematize the DT, and mitigate errors. It is also useful to explain this type of DT to people not knowledgeable with these topics. Third, the usage of low-code technologies eases the DT and allows the mitigation of related issues.

Given these conclusions, it is safe to assume that what is here proposed helps develop applications in this scope, such as a ticket system that resolves various government procedures, requested by the citizen.

6.2 Limitations and Future Work

Regarding the limitations, several were already mentioned, such as the information not going in-depth which, even if it was done on purpose since the idea was to provide a generalized view that could be adapted to other situations, meant that it was is not that relevant to the experts in the area.

Regarding the future work, an improved evaluation should be performed. As mentioned before, it was limited because it was specific to a niche field, making participants scarce, and done in a anonymous government. Also, quantitative evaluation should be performed (such as evaluating the response time or number of clicks).

ACKNOWLEDGEMENTS

This work was partially supported by the EC through project 870635 (DE4A – Digital Europe for All), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

Also, a special thank you to VILT and Prof. André for allowing the realization of this paper.

REFERENCES

Aysolmaz, B. and Demirörs, O. (2014). Deriving user requirements from business process models for automation: A case study. In *2014 IEEE 1st International Workshop on the Interrelations between Requirements Engineering and Business Process Management (REBPM)*, pages 19–28.

Dearmer, A. (2021). Complete guide to database schema design. Available: https://www.integrate.io/blog/complete-guide-to-database-schema-design-guide/. [Accessed September 10, 2022].

Kimberling, E. (2021). How amazon is a roadmap for digital transformation success. Available: https://www.thirdstage-consulting.com/amazon-roadmap-for-digital-transformation-success/. [Accessed July 13, 2022].

Liu, C. and jun tao, X. and Lu, B. (2016). Toward integrated and automated management of government affairs. *International Journal of Hybrid Information Technology*, 9:267–278.

Mazumder, A. (2020). Marreg – marriage registration [for a better cause]. In *2020 IEEE Integrated STEM Education Conference (ISEC)*, pages 1–8.

mendix (n.d.). What is low-code? Available: https://www.mendix.com/low-code-guide/. [Accessed August 19, 2022].

OpenText (n.d,a). Opentext appworks platform. Available:https://www.opentext.com/products/listing. [Accessed July 15, 2022].

OpenText (n.d.b). Opentext appworksplatform architecture. Available: https://developer.opentext.com/. [Accessed July 21, 2022].

Paul, A. and Paul, V. (2012). The e-government interoperability through enterprise architecture in indian perspective. pages 645–650.

Reijers, H.A. (2021). Business process management: The evolution of a discipline. *Computers in Industry*, 126:103404.

Russo, M. and Ferrari, A. (n.d.). Avoiding circular dependency errors in dax. Available: https://www.sqlbi.com/articles/avoiding-circular-dependency-errors-in-dax/. [Accessed July 21, 2022].

Ryan (2020). What are libraries in programming? Available: https://www.idtech.com/blog/what-are-libraries-in-coding. [Accessed August 25, 2022].