From Data Science to Modular Workflows - Changing Perspectives from Data to Platform: DBDIrl 1864-1922 Case Study

Enda O'Shea[1,2] [0000−0002−5547−9739], Marco Krumrey[1,2,3] [0009−0009−2886−5755], Daniel Sami Mitwalli[1,2,3] [0009−0005−3155−0552], Sebastian Teumert[1,2,3] [0000−0002−6483−3162], and Tiziana Margaria[1,2,3,4] [0000−0002−5547−9739]

[1] University of Limerick, Ireland {enda.oshea, krumrey.marco, mitwalli.daniel, teumert.sebastian, tiziana.margaria}@ul.ie
[2] CRT-AI: SFI Centre for Research Training in Artificial Intelligence
[3] Lero: The Irish Software Research Centre
[4] HRI: The Health Research Institute

Abstract. Many historical data collections foot on handwritten documents and registers, whose consultation is often very difficult due to the conservation state of the physical artefacts, and whose comprehension is also made difficult by the handwriting, difficult to interpret, and the language used, different from the modern terminology. Therefore significant research efforts by historians, demographers, population health scientists and others have been started in the past with the aim of making such data collections digitally available, first on the basis of images and then as readily available repositories of transcribed data in electronically queryable formats. For the purpose of extracting data from the Irish Civil registers of deaths in the DBDIrl 1864-1922 project[5], an AI-ML Data Analytics Pipeline was proposed as a working approach validated on a subset of the data. However, the pipeline requires manual steps and it is not applicable as is on similar datasets without significant modifications to its inner workings.

We are currently transforming this prototyped, single purpose product to a modular, fully automated workflow, intended to be used and reconfigured for new data in a low-code/no-code fashion by domain experts like historians. We explain our adopted analysis and refactoring process, illustrate it on part of the pipeline, including how we faced obstacles and handled pitfalls. We also evaluate its potential to become a methodical approach to transforming an interactive program to a fully automated process, in a low-code/no-code workflow style, that can be easily reused, reconfigured and extended to be able to tailor it to other datasets as needed.

Keywords: Historical data · Data science · Model driven development · Low-code/No-code · Digital Thread · DIME

[5] https://www.dbdirl.com

1 Motivation

Numerous factors can shape a programmer's coding style. In contemporary times, programmers enter the field with diverse educational backgrounds and experiences. According to a survey conducted by Stack Overflow, approximately 80% of professional programmers in 2021 have successfully obtained a bachelor's degree, leaving one in five without this qualification[6]. Concentrating on educational background, around 38.41% of programmers who held at least a bachelor's degree between 2018 and 2021 were predominantly from the field of "Computer and Information Science and Support Services"[7]. Thus, these students are often exposed to traditional programming paradigms such as object oriented programming and well established, state-of-art programming patterns. Another large share of code, however, originates from non-CS STEM fields such as engineering, mathematics, statistics, biology/medicine and business related studies. These programmers generally have been taught the bare minimum programming skills to achieve or satisfy their tasks, mostly use interpreted or scripting languages, and they operate on a paradigm where code is written down instead of properly designed, and it is seen as "disposbale" instead of a product. Other factors that influence a programmer's coding style are their level of experience, the company or organizational culture at which they have been employed, and the associated programming language(s) they used.

Further important factors that influence coding style are the purpose of the project and the role that the code plays within it. In a traditional software project, the code is used to produce a running program which is ultimately *"the product"*. In data science projects, however, scientific methods and algorithms are used to extract knowledge and insights from (possibly semi- and un-)structured data collections. In this context, the code is not primarily the product itself, but it serves as a mere tool to derive *"the product"*, which is in this case information: the output of running the code.

Being perceived as an ancillary artefact, the code involved tends to be written pragmatically and used as a short-lived and temporary tool, e.g. to explore and experiment with the data. This approach is more akin to a rapid prototyping mindset, where the code is kept pragmatic and not really curated, and its purpose is more of presenting ideas, as a proof of concept, and produce quick results. In some cases, such projects move from exploration to larger production or collaboration, where a clean and consistent style guide and well-documented code become increasingly important: that's where this prototyping style shows deficiencies, and a different mindset is required to take over.

In the DBDIrl 1864-1922 project[8], described in more detail in Section 2, there was a core need to investigate the feasibility of automating the information extraction from the Irish civil registers of deaths, provided as a large collection

[6] `https://insights.stackoverflow.com/survey/2021#education-ed-level-prof`

[7] `https://datausa.io/profile/soc/computer-programmers?employment-bl-geo=statesEBL&majors-select=commonMajors`

[8] https://www.dbdirl.com

of TIFF images page by page. This need gave rise to a "proof of concept"-style project that investigated the AI-supported extraction of information from the images and its conversion into usable data for statistical analysis. The project leaders were aware from the beginning that the medium-term goal was to produce a reusable and repurposable platform, but the starting point was from zero, with only one skilled computer scientist joining the team at a later point, with the personal aim of introducing AI and ML technologies and demonstrating that "yes, we can". The modules developed in this project are designed as interactive programs that allow the user (so far a skilled CS expert who knows AI, ML and the data very well) to follow the data processing process step by step. The results were excellent, but not transferable to other users nor datasets.

In the context of the just started GREATLEAP EU COST Action[9], however, the goal is to analyze and compare death data from historical civil registries of over 15 countries in Europe, and link them to other data collections and sources of information. In [1] and more recently in [2] we address the linkage problem of the death data with the 1901 and 1911 Irish census data. Additionally, [3] addresses coding historical causes of death data with Large Language Models and [4] introduces and discusses a common language for accessibility, interoperability, and reusability in Historical Demography. In this large scale European effort, Working Group 3 addresses the creation of analytical tools, specifically computational and visualization tools for the application to new research questions. It is clear that the process of extracting information and providing usable data is to be automated, making it easily accessible to users that are historians, and repurposable, e.g in order to handle other collections of images of datasets with "register"-like appearance. This means that the "product" shifts from the *derivation of data* and *understanding the process*, to a possibly parameterized *automated software pipeline* that effectively embodies rather complex workflows (i.e., it is neither linear nor simple). It must be able to efficiently execute these modular sub-steps and transform the provided information into usable data, covering a range of inputs and outputs.

In the following, Section 2 provides some background on the DBD-Irl 1864-1922 project and on the general class of questions such research elicits. Section 3 describes the AI-ML data analytics pipeline and its structure, then Section 4 presents the issues encountered that need resolution for a full automation. Section 5 describes the transformation needed to move to a reusable, modular application that is part of a model driven platform. Section 6 provides a comparison of the mindsets behind the two different styles of programming, on the basis of the example. Lastly, Section 7 summarizes and concludes the work.

2 DBDIrl Project Background

The "Death and Burial Data: Ireland 1864-1922" project (DBDIrl, 2017-23)[10] delved into the historical civil registration (CR) documents of Ireland to re-

[9] See https://greatleap.eu
[10] https://www.dbdirl.com

veal and analyze the biopower dynamics across this time period. From 1864 to 1922 the Irish General Register Office (GRO), operating under the authority of the British government in Ireland, systematically recorded vital life events such as births, deaths, and marriages. By thoroughly examining these records, the project aimed to reveal the dynamics surrounding life and death during this significant period in Irish history, with a particular emphasis on the conclusion of life paths in a nation marked by high emigration.

2.1 The Dataset

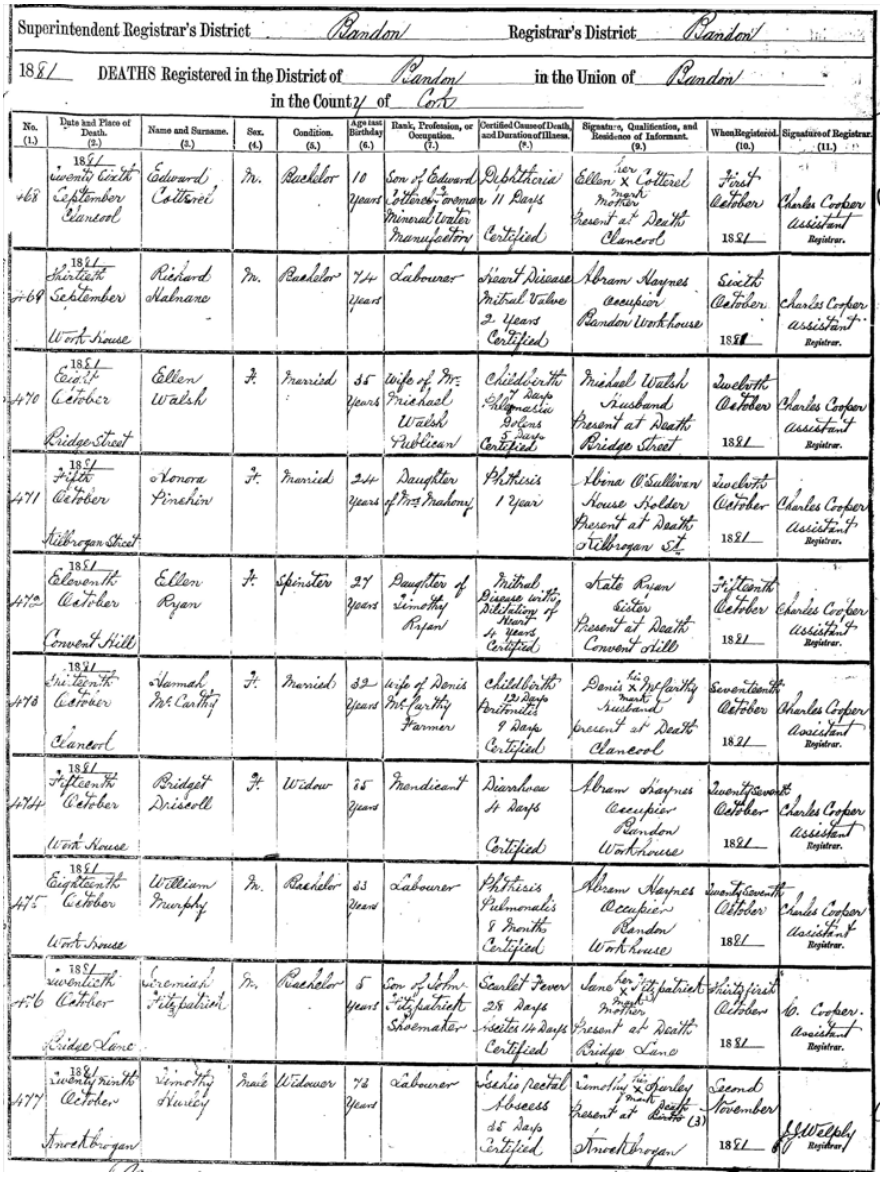
The General Register Office in Ireland facilitated the provision of CR death records from this period to the DBDIrl team, encompassing approximately 4.3 million distinct deceased individuals. These records consist of scans, as shown in Fig. 1, serving as digital reproductions of the original writings made by the registrar at the time of death, provided in both PDF and Tagged Image File Format (TIFF).

Following the partial digitization of birth, marriage, and death records by the GRO department, the accessibility and searchability of these records have been greatly enhanced on the official Irish Genealogy website[11]. Users can now explore these records using criteria such as the individual's name, location, and the timeframe of their life event. While this vast collection proves to be an invaluable resource, the digitization efforts extended to death records within the DBDIrl project demanded a more detailed approach. These records not only capture essential information about the deceased individual but also delve into details about the informant (responsible for validating the deceased's identity) and the registrar involved in the registration process. This heightened level of detail allows for a more profound understanding of the context and intricacies surrounding each documented death event. Analyzing the scanned record depicted in Fig. 1, each document can contain up to 10 individual death records, each featuring 11 columns of data. Through a collaborative effort between the DBDIrl team and computer scientists, these 11 fields were initially expanded to 22 data categories, as detailed in [5]. They were subsequently expanded to 63 categories [6] that reflect the level of granularity essential in the transcription processes for each distinct subject.

2.2 Historical Record Transcription Approaches

Manual transcription has been, and remains, a predominant approach for transcribing vast volumes of handwritten historical documents. Strategies such as crowd-sourcing [7,8], transcription workshops [9], and employing expert transcription houses [10] are the most prevalent methods for annotating large quantities of data. These approaches enable researchers to maintain control over the environments while ensuring a certain level of quality assurance. However, a clear

[11] https://www.irishgenealogy.ie/en/



Superintendent Registrar's District Bandon Registrar's District Bandon

1881 DEATHS Registered in the District of Bandon in the Union of Bandon in the County of Cork

No. (1.)	Date and Place of Death. (2.)	Name and Surname. (3.)	Sex. (4.)	Condition. (5.)	Age last Birthday (6.)	Rank, Profession, or Occupation. (7.)	Certified Cause of Death, and Duration of Illness. (8.)	Signature, Qualification, and Residence of Informant. (9.)	When Registered. (10.)	Signature of Registrar. (11.)
468	1881 Twenty Sixth September Clancool	Edward Cotterel	M.	Bachelor	10 years	Son of Edward Cotterel Overman Mineral Water Manufactory	Diphtheria 11 Days Certified	Ellen X Cotterel her mark Mother Present at Death Clancool	First October 1881	Charles Cooper Assistant Registrar.
469	1881 Thirtieth September Work House	Richard Mahanre	M.	Bachelor	74 years	Labourer	Heart Disease Mitral Valve 2 Years Certified	Abram Haynes Occupier Bandon Work House	Sixth October 1881	Charles Cooper Assistant Registrar.
470	1881 Eighth October Bridge Street	Ellen Walsh	F.	Married	35 years	Wife of Mr Michael Walsh Publican	Childbirth 17 Days Phlegmasia Dolens 12 Days Certified	Michael Walsh Husband Present at Death Bridge Street	Twelvth October 1881	Charles Cooper Assistant Registrar.
471	1881 Fifth October Kilbrogan Street	Honora Pinahin	F.	Married	24 years	Daughter of Mrs Mahony	Phthisis 1 Year	Abina O'Sullivan House Holder Present at Death Kilbrogan St.	Twelvth October 1881	Charles Cooper Assistant Registrar.
472	1881 Eleventh October Convent Hill	Ellen Ryan	F.	Spinster	27 years	Daughter of Timothy Ryan	Mitral Disease with Dilitation of Heart 4 years Certified	Kate Ryan Sister Present at Death Convent Hill	Fifteenth October 1881	Charles Cooper Assistant Registrar.
473	1881 Thirteenth October Clancool	Hannah Mc Carthy	F.	Married	32 years	Wife of Denis Mc Carthy Farmer	Childbirth Peritonitis 12 Days 9 Days Certified	Denis X Mc Carthy his mark Husband present at Death Clancool	Seventeenth October 1881	Charles Cooper Assistant Registrar.
474	1881 Fifteenth October Work House	Bridget Driscoll	F.	Widow	75 years	Mendicant	Diarrhoea 4 Days Certified	Abram Haynes Occupier Bandon Workhouse	Twenty Second October 1881	Charles Cooper Assistant Registrar.
475	1881 Eighteenth October Work House	William Murphy	M.	Bachelor	83 years	Labourer	Phthisis Pulmonalis 8 Months Certified	Abram Haynes Occupier Bandon Work House	Twenty Seventh October 1881	Charles Cooper Assistant Registrar.
476	1881 Twentieth October Bridge Lane	Veronica Fitzpatrick	M.	Bachelor	5 Years	Son of John Fitzpatrick Shoemaker	Scarlet Fever 28 Days Diarrhoea 14 Days Certified	Jane Lee X Fitzpatrick her mark Mother Present at Death Bridge Lane	Thirty first October 1881	H. Cooper. Assistant Registrar.
477	1881 Twenty ninth October Knockbrogan	Timothy Hurley	Male	Widower	78 Years	Labourer	Diarrhoea Pectoral Abscess 35 Days Certified	Timothy Hurley his mark Present at Death Knockbrogan	Second November 1881	J. J. Wolfely Registrar.

Fig. 1. Example of Scanned CR Death Record from 1881

challenge arises as the time and resources available to different researchers may vary.

Looking at automated transcription systems in the study by Thorvaldsen et al. [7], the researchers utilized two distinct versions to handle both historical Norwegian census (1891) and Barcelona marriage records (1451 to 1905), with each system catering to different layouts and data type representations. The Norwegian census data were presented in a uniform, structured document, enabling the utilization of anchor regions for layout analysis and the isolation of word regions based on these anchors. This process was executed using open-source tools and libraries, resulting in the identification of 99.3% of underlined fields within the document and the correct classification of 96.7% of these underlines. Subsequently, they segmented data regions into individual images, achieved up to 70% accuracy in classifying single digits, and generated name clouds by gender to aid future manual transcription endeavors.

Conversely, the Barcelona marriage records were recorded as lines of sequential handwritten text, necessitating additional segmentation compared to the Norwegian census data. The team developed a three-tiered, web-based, crowdsourced platform, gaining access to over 150 transcribers who annotated two years' worth of data and provided essential ground truth data for image analysis. Following this, a handwriting recognition system was integrated to support the transcribers during the manual transcription phase. The handwriting recognition module places a focus on the segmentation of text blocks into individual paragraphs using probability maps to identify textual features. Subsequently, these paragraphs are further segmented into singular lines through the application of graph-based continual path measures.

Thorvaldsen subsequently extended this research [11] by utilizing 353,000 commercially annotated images. These images were employed to train a deep learning network with the goal of classifying first names listed in the Norwegian 1950 census, yielding the ten most probable results for each name classification. Focusing on the Norwegian 1950 census, [12] furthered this research by creating the 'Occode' end-to-end machine transcription pipeline. This pipeline was designed to transcribe 2.3 million handwritten three-digit occupation codes embedded within the census data. To capture the occupation codes, an ad hoc segmentation program was applied, and multiple processing methods for the codes were tested. The sequence model demonstrated outstanding performance, achieving an automated transcription accuracy of 97%. The residual 3% of the data was subjected to manual transcription, underscoring once again the necessity for a hybrid approach in handling such handwritten data.

In automated transcription pipelines, such as those outlined above, a key recurring theme is the necessity to segment various types of documents into smaller, more detailed data units, such as table cells or individual words in a text line. This segmentation is crucial for the effective processing of these documents by language models. Only after this segmentation can tasks like word spotting, word classification, or character-level classification be effectively undertaken. Additionally, the integration of lexicons is a common practice in these

pipelines, as they considerably narrow down the search space for word classification, playing a vital role in achieving desirable results.

This approach is exemplified in [13], [14], [15], and [16], where vertical and horizontal projection filters, skew adjustments followed by 2D mask generation, and the use of anchor points from OCR transcription software are utilized to identify cell regions. More modern methods that incorporate deep learning, including semantic and instance segmentation, are used in [17], [18], and [19]. These sophisticated strategies emphasize the automated extraction of features followed by the identification and extraction of crucial regions of interest. Subsequently, classification networks and techniques can be applied to these extracted elements.

2.3 Manual Transcription in DBDIrl

In the early phases of the DBDIrl project, the data underwent manual transcription by domain experts, specifically members of the DBDIrl team. Despite being a painstaking and time-intensive procedure, it yielded high-quality results, ensuring the accuracy needed to conduct quantitative research. Challenges emerged, however, when attempting to expand this approach to include non-domain experts, such as students, and special interest groups. These new stakeholders necessitated extensive training to accurately extract and transcribe documents with the required level of granularity, adhering to strict guidelines to steer their progress. These guidelines were established through the development of a dedicated Web application, the Historian DIME App (HDA) [9], featuring individual pages that guide the transcription of records based on a page for the information provided by each role (registrar, informant, deceased) involved in the registration of that death. The software itself was developed using the eXtreme Model Driven Development approach (XMDD) [20]. The transcription covered all 63 data elements, equipped with predefined correctness checks (expressed by syntactic and semantic constraints) that ensured correctness and plausibility of the input data. Furthermore, a set of classifiers was developed [6] to identify errors in previously transcribed data, initially captured in Excel sheets. The classifiers analyzed the data record by record and field by field, providing feedback on each entry as well as aggregated percentages of accurately transcribed sets for the fields pertaining to each of the three roles.

2.4 Automating the Transcription though an AI/ML Pipeline

While the manual transcription process demonstrated a commendable level of precision, the sheer volume of over 4 million individual death records warranted a move towards automation. Given the monumental scale of processing, involving nearly 60 years' worth of data, each comprising approximately 12,000 scanned documents, it became imperative to adopt a systematic pipeline approach. This method entailed a series of sequential steps: initially, the documents were meticulously narrowed down to table regions containing death records, then further refined to individual row and column regions. Subsequently, the focus shifted

to the identification and classification of individual words within these refined regions.

The decision to segment the documents only at the word level was influenced by the handwritten cursive nature of the texts, which posed significant challenges for character segmentation. Moreover, the lexicon within multiple column regions was limited to specific domains, simplifying the word-level classification process. To streamline the workflow, the pipeline was constructed as a series of distinct modules, each dedicated to a specific aspect of the transcription process. Each stage of the pipeline was tasked with processing an entire year's worth of data before passing it on to subsequent stages. The advantage of this approach is that upon completion of a year's worth of data, and its quality have been assured, the outputs could then be integrated into the training sets for the machine learning models, ensuring consistent updates and improving transcription accuracy as the project expanded.

The central objective was approached from a data science standpoint, aiming to assess the feasibility and overall capability of the project. Although the project was developed in separate modules, each targeting specific concerns with varied techniques, the ultimate goal remained to create a unified product capable of seamless operation throughout its entirety. This automated end-to-end process must effectively manage both yearly volumes of data and individual distinct records. At present, several endpoints within the pipeline necessitate manual intervention, whether it's for data inspection or processing to enable subsequent stages to commence. The forthcoming sections will delve into these limitations and outline the prerequisites for fully automating the pipeline, thereby eliminating the need for such manual interventions.

3 The AI-ML Data Analytics Pipeline

Figure 2 illustrates the implicit flow graph underlying the modules of the automated data analytics pipeline. It is a linear pipeline, where the output information produced by one stage is passed on to the successive module for further processing. It takes images provided as TIFF files as input and starts with the recognition of the shape of the table within these images through the use of image segmentation, followed by the extraction of said table.

After extracting a table image, the rows containing individual deceased records undergo segmentation and extraction in the *Row Segmentation* stage. Following this process, for each individual row its 11 columns are segmented in the *Column Segmentation* stage, providing access to the single cells. This top-down approach decomposes the structure of the original TIFF file image to its finest granular level, enabling the subsequent content analysis, through identification of each individual word for each cell. This approach is very general, and it is applicable to all document types containing the same tabular structures. This enables a wide reusability, and it can lead to processing at scale using these procedures.

Within these cell regions, the words and phrases can now be identified using object detection networks in the *Word Detection* stage of Figure 2. Depending on

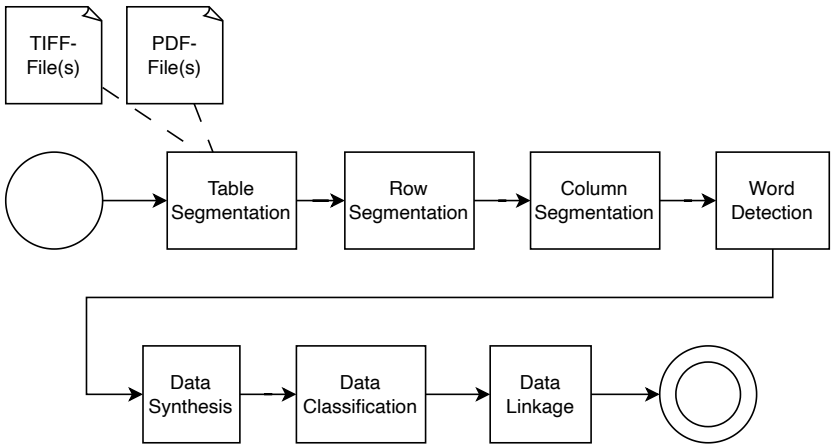


Fig. 2. Flow graph underlying the logic flow of the prototype in [21].

the cell region and expressiveness of the lexicon within this region, data synthesis may then be used to create training data for a series of classifiers (*Data Synthesis* stage). Finally, models are trained and data are classified (*Classification* stage) to generate a normalization of the resulting data, to counteract the subjective nature of human formulation in writing.

Disregarding aspects such as data annotation and model training, a closer examination of this flow graph reveals a simple, unidirectional data flow, suggesting that, in theory, it could be readily transformed into an automated process. However, if we look for example at the internal organization of the *Table Segmentation* module (see Fig. 3), we can recognize some obstacles.

3.1 The Complex *Table Segmentation* Module

The *Table Segmentation* module is the initial step in the pipeline that takes image files of tables in the form of TIFF files as input but is not actually an atomic step: it has a complex internal structure with multiple sub-modules. As shown in Fig. 3, there are two main consecutive steps: the *Project Preparations* and the actual *Table Segmentation*.

3.2 The *Project Preparations* Submodule

The *Project Preparations* submodule consists itself of three parallel steps that prepare the projects input data.

– Before any operation, the *Project Folder Setup* program establishes a folder structure utilized by the other modules in Fig. 2. This setup is performed only once per year of data and ensures a consistent structure throughout

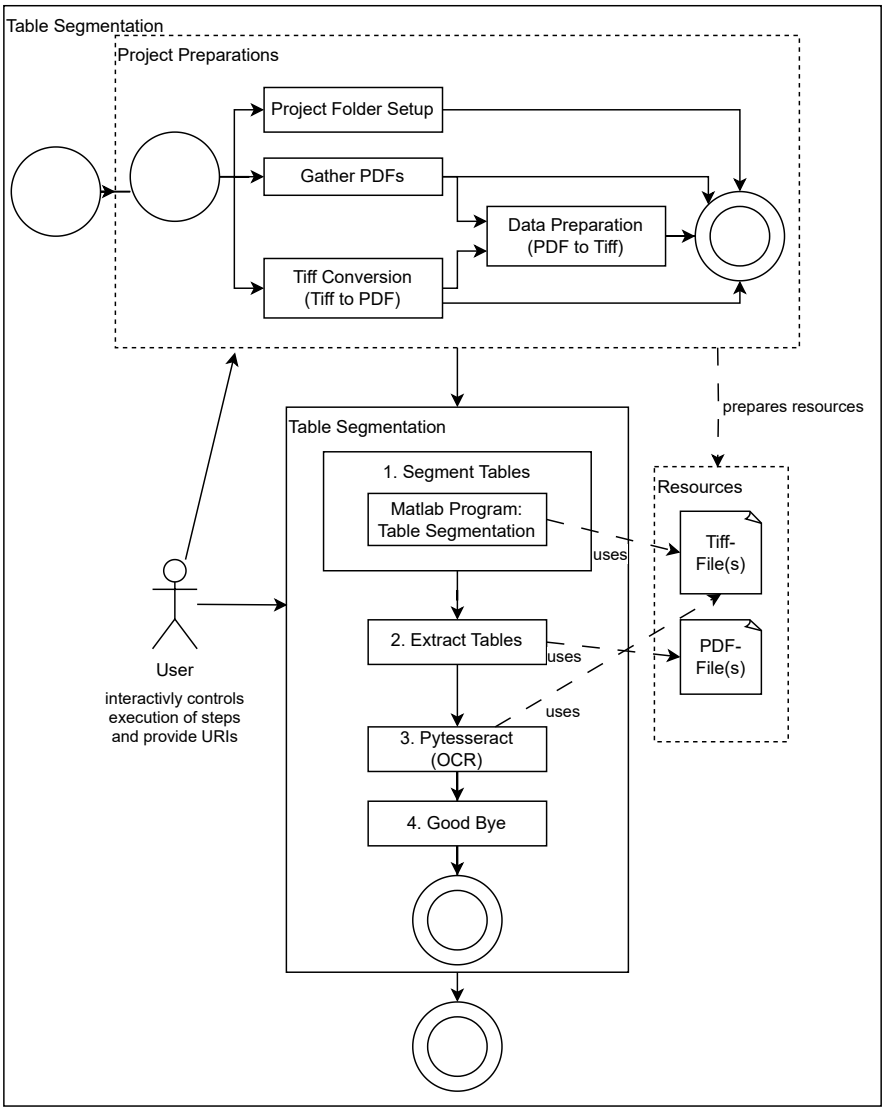


Fig. 3. A deeper look at the functionality of the *Table Segmentation* module of Section 2.

the project. This consistency enables the utilization of static location-based arguments when processing artifacts such as image and text files.
– The data provided to the DBDIrl team comprised numerous images in PDF and TIFF file formats, organized annually and by district. Within these folders, around 20% were in PDF format, while the majority, approximately 80%, were in TIFF format. A decision regarding the preferred data type for further processing needed to be made. TIFF files were selected due to their recognized capability to store high-quality images with a resolution of 300 Dots Per Inch (DPI), ensuring optimal detail. This is especially crucial for classifying handwritten text images. This choice is particularly significant as the focus involves narrowing down the document scope to individual words, emphasizing the importance of preserving high-quality image types.
However, once again, the process of constructing a dataset of TIFF files is not straightforward and necessitates additional handling. The TIFF files supplied by the GRO vary in size and have a bit depth level of 1 (i.e., just black and white), posing potential issues in the table segmentation module. This is because the network for this module requires input with 3 RGB channels and a consistent size structure. Therefore, there is a need for processing the TIFF files at scale.
The most straightforward approach involves adding additional channels to the original TIFF image. However, this method resulted in blurring and a reduction in quality [21]. To address this issue, all available TIFF files were first converted to PDF format, unifying the dataset to 100% PDF. These PDFs were then converted back to the final TIFF format with 3 RGB channels. While the ideal size of the images would be [6400x4800x3], this caused memory issues with certain AI network types. It was then determined that the input to the network could be 1/16 of this size and still operate adequately. This process ensured the required input format for the subsequent stages of the data analysis pipeline while preserving the necessary quality.

Note that this process involves user interaction: the user must select the folders containing the specified images to be processed. Once all the data is converted into the conforming formats, it becomes ready for use to extract the tables from the images: the *Table Segmentation* in Fig. 3.

3.3 The Actual *Table Segmentation* Submodule

The submodule *Table Segmentation* is also realized as an interactive program. It gives the user the possibility to execute all its actions in the Phases 1 to 4 as desired, even if they should be executed consecutively.

1. In Phase 1, the *Segment Tables* action carries out the actual segmentation of the tables. This is achieved by initiating a child process implemented in Matlab, i.e. as a platform-dependent program on the Windows operating system. The process prompts the user to specify the location of the TIFF images whose table information needs segmentation. The segmentation utilizes

a deep neural network model that identifies the coordinates of the specific *region of interest* representing a table. It outputs results as key-value pairs where the key is the absolute file path of the image's location, and the value is an array with the coordinates of the four points defining the table's location inside the image. These coordinate values must be scaled by a factor of 16 when performing extraction.

2. The *Extract Tables* stage performs the actual extraction by cropping the segmented regions out of the images. This is done on a page by page basis, where a PDF file is first converted into a TIFF image with an ideal size of [4800x6400x3]. The *regions of interest* identified by the output of *Segment Tables* and scaled by a factor 16 are then extracted. Since the death records were scanned from a physical register source, their orientations and ranges may vary from page to page. To address this, a de-skewing process is employed before applying the Lanczos interpolation method, all aimed at ensuring the standardization of the resulting table image at a size of [3600x4800x3]. This table image is not saved to disk, rather it is immediately sent into the Row Segmentation module for separation. Upon successfully extracting the table region and performing the row segmentation operation, the original [4800x6400x3] image is deleted from memory before processing the following page.
3. The *Pytesseract* action applies the Pytesseract Optical Character Recognition (OCR) tool [**?**] to the extracted table region images. This serves as a qualitative assessment, ensuring the quality of the extracted table regions by verifying the presence of the printed word *'Superintendent'* in the appropriate region, as depicted in the top-left position of Fig. 1. By creating a set region box across this area, serving as an anchor point, an Intersection over Union test is performed to compare the location of the ideal region with the resulting bounding box produced by the Pytesseract system. If the Intersection over Union falls below a specified threshold when applied across a subset of a year's data, it indicates an issue in the Table Segmentation process. Consequently, the pipeline cannot proceed until the problem is addressed.
4. If the all the tests have passed and the user is satisfied with the outcome, the user can close the program by selecting *Good Bye* to complete the module *Table Segmentation* and exit.

4 Issues Encountered for Full Automation

Looking at the internal structure and characteristics of this module, one can notice some pitfalls and obstacles that complicate the transformation of the interactive module into a fully automated process.

1. **No Interactivity / Convention over Configuration**: Each sub-module of the given module prompts the user for some input (e.g., file paths) or to manually execute parts of the process. This interactivity is useful when designing the pipeline step by step, exploring the data, prototyping ideas, and

learning to understand the proposed procedure of the *Machine Transcription Pipeline* of [21]. To automate the procedure, however, this interaction is disruptive and must be removed and replaced by agreed conventions. While a folder structure that serves such a conformance is created in the *Project Folder Setup*, it seems to be more suited for the other modules seen in Fig. 2 and its root needs to be specified manually by the user.

2. **Separation of Concerns**: In data science projects, scientific methods and algorithms are used to extract knowledge and insights from data that is often semi-structured or unstructured, like in our case the images of register pages. The code merely serves as a tool to derive information, and it is not seen as a primary product itself, as is customary in software engineering projects. The clean organization of the code is frequently affected. The *Project Folder Setup* sub-module establishes a folder structure that affects the whole procedure seen in Fig. 2. As it covers a much larger scope than where it is located, two nesting levels under the top-level pipeline, it should not be part of this module, but instead constitute itself a new module, initialising all subsequent stages in the Fig. 2 flow graph.
3. **Data Conformance and Optimization**: The *Table Segmentation* sub-module expects two file formats, TIFF and PDF, each chosen for specific purposes. Upon conversion from PDF to TIFF file images, the size is considerable at $[4800x6400x3]$ with a 300 DPI. The TIFF image employed in the sub-module is a scaled down $[300x400x3]$ version, which is only used for segmentation by a neural network (see *Matlab Program: Table Segmentation* in Fig. 3) to rapidly determine the corner points. The sub-module *Extract Tables* expects the PDF files containing the original information, which are converted into temporary full sized TIFF images (see Figure 3), which are then used to extract the region of interest. Such a converted TIFF image takes up a considerable amount of memory and does not scale well with the number of PDF files that need to be processed: just this experimental subset of data consists of 2000 TIFFs at 300DPI with a resolution of $4800x6400x3$ taking approximately 176 GB of storage space. Therefore, only a limited number of files can be processed in parallel, after which they are deleted to recover storage.

   The choice of file formats to be used should be evaluated to optimize the performance of the whole process. In this case, the number of file conversions and the size of the data should be reduced to increase the number of files that can be processed in parallel. This could be achieved by finding a more appropriate file format for this particular problem domain. A good candidate for this case could be PNG. While both TIFF and PNG are lossless, PNG has a reduced file size due to compression by limiting the number of color spaces supported.
4. **Remove Dead Code**: The action *Pytesseract* in the *Table Segmentation* sub-module is an ancillary procedure to perform qualitative assessment upon completion of the Table Segmentation process, but it is irrelevant in the scope of the automated pipeline shown in Fig. 2.

Additionally, a series of modifications and transformative methods are repeated throughout the pipeline. These would need to be refactored into a single process to avoid code redundancy and enhance maintainability.

5. **Platform Independence**: The regions of interest inside the images are identified by a Matlab program executed by the *Segment Tables* process (see Phase 1: *Matlab Program: Table Segmentation* in *Segment Tables*). In an early version of this project, the Matlab program was platform-dependent on the Windows operating system. Initially, the pipeline operated on a file by file basis, employing the executable to perform segmentation when required. This approach restricted the executability (at least) for this module to only one operating system/platform. To give better support for a possible migration to other platforms, such a procedure should be executed in a platform-independent way. Fortunately, the Matlab runtime environment offers the possibility to run the Matlab program as a script, platform-independent. This functionality was not used in this project, but rather the explicit execution of a Windows executable of the Matlab program from within the Python code.
   It should be noted that this singular processing approach was later changed to operate at scale by writing all segmentation elements to text files at once, avoiding the repeated usage of the Matlab executable [6].

This analysis concerns only the first module of the flow graph in Figure 2, therefore the issues identified so far might not be the complete set of problems to be considered when migrating the program based on a data science project to an automated process: we expect to find more kinds and variety of migration problems. They are however indicative of the spread of concerns that play a role when moving towards a "platform" thinking, with robust, efficient and easily reusable modules, which is our next goal.

5 From a Single-Purpose Monolith to General-Purpose Modules

The workflow as outlined above is a custom tailored collection of scripts geared towards the specific use-case of these death-and-burial records and it works around the idiosyncrasies, problems and oddities that are specific to these kinds of documents. However, many of these problems are of a general nature. By swapping just parts of the workflow, it should be possible to adapt the workflow to other sets of register-like sources, extending the applicability to problem domains with different kinds of records with different kinds of information. Furthermore, automating the workflow should allow batch processing of the whole collection of all death-and-burial records, especially for all years, without human intervention. Note that this way there is no restriction on the transcription to a specific language if there are suitable trained models for other languages.

Thus, there are two independent drivers for automating the workflow. In the short term, automating the workflow would quickly generate more usable

results for other years of the same dataset, that can be used by Irish historians to advance their research interests. Automation would also pave the way for improvements in modularization and (re-)composition.

However, the pipeline as developed is monolithic in nature, with many implicit dependencies between the steps, which require careful manual orchestration. There are many interaction points in the current work, which require human intervention and/or input at the time of execution. For a full automation it would be necessary to remove these interaction points and instead collect the required inputs as one input vector that can be provided ahead-of-time, prior to execution of the workflow. This can be implemented as a configuration file, command-line parameters or any other form of parameterization from which these values can be collected and then injected at the proper points during runtime.

Similarly, a desirable property for each step is purity. Ideally, a single step would only consume a defined set of inputs and generate a defined set of outputs. While this is a desirable property in theory, practice shows that it is not always feasible. Nonetheless, side-effects onto the runtime-environment, the system or data should be minimized and formally captured in an appropriate representation that allows for this data to be formally reasoned about and to be tractable for a tool that automatically orchestrates the workflow. We have found that many steps in the current workflow leverage implicit dependencies on the state of the environment and in-memory state, and manipulate state via side-effects, i.e. through global variables that change during loop iteration. This does not pose a problem if the steps are executed one-by-one, as single processes, but it severely restricts the ability for automated tools to freely re-compose the steps into single scripts that have a shared runtime lifecycle.

In a similar vein, redundancies exist in the current workflow where similar code occurs in multiple places. By treating the whole workflow as a model, these redundancies can and should be eliminated and transformed into components that can be re-parameterized and re-used at various locations, and then woven into the finished workflow by means of an automated orchestrator.

Aside of purity or the formal capture of side effects, idempotency [22] (and, related to that, binary reproducibility) is another desirable property for any such workflow. While purity is not practically attainable in many cases, idempotency is. In this context, an idempotent operation may have side-effects (it is not pure), but given the same inputs, the same operation always produces the exact same side effects (for example, generating the same files with the same content). Thus, idempotency can be leveraged for caching, cache reconstruction and many other desirable features.

The current workflow is platform specific and only works on a Windows-platform. It also has hard-coded mitigation strategies for some limitations of that platform. Removing some of these platform dependencies is rather trivial, for example replacing calls to Windows `.exe` files with generic command-line calls or even moving to a containerized execution. Other platform-specific aspects, for example to combat limitations on the size of file handles, need more careful consideration. Again, a model-based approach could be leveraged to auto-

generate this code for each relevant platform, instead of requiring it to be baked into the user-level code.

A model-based approach would also allow for better scheduling. The workflow as presented is linear in nature, as it is executed by a human step-by-step. But, if viewed from an information science point of view, it represents a dataflow that can be modeled as directed-acyclic-graph. It forms a dependency-tree with multiple fork- and join-points, and has several paths that could be executed in parallel and then synchronized at the join points. An orchestrator could freely rearrange the execution steps to improve execution times by parallelizing execution and smart caching of dependencies.

In summary, redefining the workflow as a proper process model and reshaping the scripts from the prior work into pure, idempotent, clearly defined and interaction-free modules would enable a great deal of optimization techniques to be applied. Modelling the workflow as dataflow would also enable security and quality assurance aspects to be checked statically, for example with model checking [23]. It is also an essential step in providing a low-code/no-code graphical environment to freely re-compose the workflow. Such a graphical mIDE [24,25] can enable non-experts to better understand and even change parts of the workflow [26]. Automating the workflow would furthermore allow interfacing with existing low-code/no-code tools that are already used in the field today, e.g. the Historian DIME App [9]. Thus, it would make the prior work much more accessible and usable for Irish historians and enable even closer cross-domain and interdisciplinary collaboration between historians, data scientists and computer scientists.

6 Analytical Overview

After detailing the challenges and issues present in the current state of the pipeline in Section 3, and outlining the requirements and general approach to transforming it into general-purpose modules, this section aims to provide an assessment of the individual pipeline steps with respect to their current state regarding the individuated obstacles and pitfalls. We look at three major categories of quality for the outcome of the automation process:

- **Maintainability / Usability**: This category describes problems such as *Separation of Concerns* and *Dead Code* [27] but also incoherent code style in general. While these problems may seem a minor concern, they should be taken into consideration when rebuilding the pipeline as they impact the future maintenance and usability of the platform.
- **Performance**: Efficiency is a significant concern, particularly in the realm of data science where handling substantial amounts of data is common. This category encompasses issues stemming from data inconsistencies, redundancies in processing steps, and broader performance concerns such as unnecessary cyclomatic complexity.

– **Automation alignment**: The final category comprises problems directly affecting the transformation of the pipeline into an automated process. Current instances of interactivity, such as prompting for data input and configuration settings, must be either fulfilled by injecting the necessary information or entirely replaced by predefined configurations. Additionally, all process steps should be platform-independent to remove restrictions on supported operating systems.

Table 1. Occurrences of issues in each pipeline segment, per category

	Maintainability	Performance	Alignment
Table Segmentation	3	1	2
Row Segmentation	1	0	1
Column Segmentation	2	2	2
Word Detection	0	0	1
Data Synthesis	0	0	1
Data Classification	4	2	5
Total	10	5	12

Table 1 provides an overview of the number of major technical issues that require immediate remedying within each pipeline module. The numbers differ across the categories.

– *Performance* issues are the least observed, with 5 instances. This is positive, as it indicates that the initial purpose of the pipeline, the efficient extraction of data, is already quite satisfied.
– *Maintainability* issues (10 instances) are mostly due to the prototyping nature and thus quickly changing code base.
– *Alignment* issues are the most frequent, with 12 instances. This was to be expected, because the transformation into an automated process was not in the initial scope of the project and thus of no concern until now.

7 Conclusion

We have shown that there is a big difference between the code organization and the orchestration choices in a typical data analytics and AI based project, aimed at solving a specific problem for experts, and the organization needed for a platform based reusable, extensible domain specific implementation that makes the single components platform-agnostic, manages dependencies in an organised, separate way. Similar approaches have been successful in the past for generic tool wrapping [28], in applications to, e.g., bioinformatics [29] and climate change analysis [30], and in Semantic Web applications [31]. This approach supports an easy reuse and repurposing of the components, together with an ease of evolution

and reorganization of the processes, substituting the hardwired pipeline or highly interactive manual execution.

The next steps will transform the pipeline into an automated process. This reconstruction and transformation is the major goal going forward, and in this course several concerns must be addressed. First, all manual interactions need to be served either by externally injecting needed information or by predefined configurations. Furthermore, all the current instances of platform-dependent process steps need to be replaced by platform-independent operations.

Besides these, a major concern lies in the way big data sets and models are processed. Transferring data between independent process steps needs to be avoided as it causes unnecessary overhead regarding performance and traffic load. Thus, data should be processed in place.

So far the main data transcription was done with the Historian DIME App [32], which enables a fast but manual input of transcribed text or data. As a first step towards automation, this app should be enhanced with the content recognition workflow in order to pre-fill automatically identified data into the corresponding fields. This way the users could either confirm (if correct) or otherwise amend the prefilled data, and save time while providing a better training data for the automatic recogniser.

As we have seen in the GREATLEAP context, the sheer volume of TIFFs available to archives, collections, and online is astounding. Considering that their high quality transcription is a once-only activity, and that it is essential to unleash a wealth of research relevant to many disciplines, it is definitely worth to research how to systematically produce excellent AI and ML supported capabilities for fully or semi-automatic transcription.

We intend to do it in an XMDD and LC/NC fashion as such a platform has the best chances to leverage the community of practices and empower the researchers to "self-assemble" or evolve and modify workflows for their data sources and aims.

This modular workflow has huge potential not just for historians: government documents, handwritten patient records in hospitals, old business records, legal records and other documents exist in large numbers, and they are unused or hardly accessible. Eventually, the goal is to achieve a fully configurable graphical transcription environment that can quickly adapt the transcription process to other types of documents.

Acknowledgements

This work was conducted with the financial support of Science Foundation Ireland (SFI) under grants number 18/CRT/6223 574 (SFI Centre of Research Training in AI) and 21/SPP/9979 (R@ISE).

References

1. Adam J. Doherty, Rachel A. Murphy, Alexander Schieweck, Stuart Clancy, Ciara Breathnach, and Tiziana Margaria. Censusirl: Historical census data preparation with mdd support. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 2507–2514, 2022.
2. Rachel A. Murphy, Ciara Breathnach, Alexander Schieweck, and Tiziana Margaria. Interoperating civil registration of death and census data: Old age and marriage as categories of analysis. In *Proc. AISoLA 2023*. Springer, 2024.
3. Bjørn-Richard Pedersen, Maisha Islam, Doris Tove Kristoffersen, Lars Ailo Bongo, Eilidh Garrett, Alice Reid, and Hilde Sommerseth. Coding historical causes of death data with large language models. In *Proc. AISoLA 2023*. Springer, 2024.
4. Rick J. Mourits, Tim Riswick, and Rombert Stapel. Common language for accessibility, interoperability, and reusability in historical demography. In *Proc. AISoLA 2023*. Springer, 2024.
5. Ciara Breathnach, Najhan M Ibrahim, Stuart Clancy, and Tiziana Margaria. Towards model checking product lines in the digital humanities: An application to historical data. In *From Software Engineering to Formal Methods and Tools, and Back*, pages 338–364. Springer, 2019.
6. Enda O'Shea, Rafflesia Khan, Ciara Breathnach, and Tiziana Margaria. Towards automatic data cleansing and classification of valid historical data an incremental approach based on mdd. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1914–1923, 2020.
7. Josep Lladós, Alicia Fornés, Anna Cabré, Trygve Andersen, Gunnar Thorvaldsen, Joana Maria Pujadas-Mora, and Line Eikvil. A Tale of Two Transcriptions Machine-Assisted Transcription of Historical Sources. 2015.
8. archives.gov. 1950 US Census Transcription Projects, 2023. Accessed: 2023-09-25.
9. Alexander Schieweck, Rachel Murphy, Rafflesia Khan, Ciara Breathnach, and Tiziana Margaria. Evolution of the historian data entry application: Supporting transcribathons in the digital humanities through mdd. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 177–186, 2022.
10. findmypast.co.uk. How was the 1921 census transcribed, 2023. Accessed: 2023-09-25.
11. Gunnar Thorvaldsen. Automating Historical Source Transcription. *Historical Life Course Studies*, 10:59–63, March 2021.
12. Bjørn-Richard Pedersen, Einar Holsbø, Trygve Andersen, Nikita Shvetsov, Johan Ravn, Hilde Leikny Sommerseth, and Lars Ailo Bongo. Lessons Learned Developing and Using a Machine Learning Model to Automatically Transcribe 2.3 Million Handwritten Occupation Codes. *Historical Life Course Studies*, 12:1–17, January 2022.
13. Joan Mas, Alicia Fornés, and Josep Lladós. An Interactive Transcription System of Census Records Using Word-Spotting Based Information Transfer. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 54–59, 2016.
14. Cédric Sibade, Thomas Retornaz, Thibauld Nion, Romain Lerallut, and Christopher Kermorvant. Automatic Indexing of French Handwritten Census Registers for Probate Geneaology. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, page 51–58, New York, NY, USA, 2011. Association for Computing Machinery.

15. Alessandra B. S. Almeida, Rafael Dueire Lins, and Gabriel de F. Pereira e Silva. Thanatos: Automatically Retrieving Information from Death Certificates in Brazil. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, page 146–153, New York, NY, USA, 2011. Association for Computing Machinery.
16. Thibauld Nion, Farès Menasri, Jérôme Louradour, Cédric Sibade, Thomas Retornaz, Pierre-Yves Métaireau, and Christopher Kermorvant. Handwritten Information Extraction from Historical Census Documents. In *ICDAR*, pages 822–826, 2013.
17. Christian M. Dahl, Torben S. D. Johansen, Emil N. Sørensen, Christian E. Westermann, and Simon F. Wittrock. Applications of Machine Learning in Document Digitisation, 2021.
18. Weihong Lin, Zheng Sun, Chixiang Ma, Mingze Li, Jiawei Wang, Lei Sun, and Qiang Huo. TSRFormer: Table Structure Recognition with Transformers, 2022.
19. Rémi Petitpierre, Marion Kramer, and Lucas Rappo. An end-to-end pipeline for historical censuses processing. *International Journal on Document Analysis and Recognition (IJDAR)*, March 2023.
20. Tiziana Margaria and Bernhard Steffen. *Service-Orientation: Conquering Complexity with XMDD*, pages 217–236. Springer London, London, 2012.
21. Enda O'Shea. Machine transcription pipeline for death and burial data - ireland 1864-1922 (dbdirl). 2024.
22. Benjamin Peirce. Linear associative algebra. page 97–229, 1881.
23. Bernhard Steffen. Data Flow Analysis as Model Checking. In *Proceedings of the International Conference on Theoretical Aspects of Computer Software*, pages 346–365. Springer-Verlag, 1991.
24. Bernhard Steffen, Frederik Gossen, Stefan Naujokat, and Tiziana Margaria. Language-Driven Engineering: From General-Purpose to Purpose-Specific Languages. In Bernhard Steffen and Gerhard Woeginger, editors, *Computing and Software Science: State of the Art and Perspectives*, volume 10000. Springer, 2019.
25. Michael Lybecait, Dawid Kopetzki, Philip Zweihoff, Annika Fuhge, Stefan Naujokat, and Bernhard Steffen. A Tutorial Introduction to Graphical Modeling and Metamodeling with Cinco. In *Proc. of the 8th Int. Symp. on Leveraging Applications of Formal Methods, Verification and Validation, Part I Modeling (ISoLA 2018)*, volume 11244, pages 519–538. Springer, 2018.
26. Anna-Lena Lamprecht. *User-Level Workflow Design - A Bioinformatics Perspective*, volume 8311 of *Lecture Notes in Computer Science*. Springer, 2013.
27. Simone Romano, Christopher Vendome, Giuseppe Scanniello, and Denys Poshyvanyk. A multi-study investigation into dead code. *IEEE Transactions on Software Engineering*, 46(1):71–99, 2020.
28. Tiziana Margaria. Web services-based tool-integration in the eti platform. *Software & Systems Modeling*, 4:141–156, 2005.
29. Anna-Lena Lamprecht, Tiziana Margaria, and Bernhard Steffen. Seven variations of an alignment workflow-an illustration of agile process design and management in bio-jeti. In *Bioinformatics Research and Applications: Fourth International Symposium, ISBRA 2008, Atlanta, GA, USA, May 6-9, 2008. Proceedings 4*, pages 445–456. Springer, 2008.
30. Samih Al-areqi, Anna-Lena Lamprecht, Tiziana Margaria, Steffen Kriewald, Dominik Reusser, and Markus Wrobel. Agile workflows for climate impact risk assessment based on the ci: grasp platform and the jabc modeling framework. 2014.

31. Charles J Petrie, Tiziana Margaria, Holger Lausen, and Michal Zaremba. *Semantic Web Services challenge: Results from the first year*, volume 8. Springer Science & Business Media, 2008.
32. Alexander Schieweck, Rachel Murphy, Rafflesia Khan, Ciara Breathnach, and Tiziana Margaria. Evolution of the historian data entry application: Supporting transcribathons in the digital humanities through mdd. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 177–186, 2022.