Electronic Communications of the EASST
Volume 82 (2022)

11th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation

-

Doctoral Symposium, 2022

Low-Code/No-Code Artificial Intelligence Platforms for the Health Informatics Domain

Colm Brandon,Tiziana Margaria

23 pages

Guest Editors: Sven Jörges, Salim Saay, Steven Smyth
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Low-Code/No-Code Artificial Intelligence Platforms for the Health Informatics Domain

Colm Brandon[1]*,**Tiziana Margaria**[2]

[1] colm.brandon@ul.ie
University of Limerick, Limerick, Ireland

[2] tiziana.margaria@ul.ie
University of Limerick, Limerick, Ireland

Abstract: In the contemporary health informatics space, Artificial Intelligence (AI) has become a necessity for the extraction of actionable knowledge in a timely manner. Low-code/No-Code (LCNC) AI Platforms enable domain experts to leverage the value that AI has to offer by lowering the technical skills overhead. We develop domain-specific, service-orientated platforms in the context of two subdomains of health informatics. We address in this work the core principles and the architectures of these platforms whose functionality we are constantly extending. Our work conforms to best practices with respect to the integration and interoperability of external services and provides process orchestration in a LCNC model-driven fashion. We chose the CINCO product DIME and a bespoke tool developed in CINCO Cloud to serve as the underlying infrastructure for our LCNC platforms which address the requirements from our two application domains; public health and biomedical research. In the context of public health, an environment for building AI driven web applications for the automated evaluation of Web-based Health Information (WBHI). With respect to biomedical research, an AI driven workflow environment for the computational analysis of highly-plexed tissue images. We extended both underlying application stacks to support the various AI service functionality needed to address the requirements of the two application domains. The two case studies presented outline the methodology of developing these platforms through co-design with experts in the respective domains. Moving forward we anticipate we will increasingly re-use components which will reduce the development overhead for extending our existing platforms or developing new applications in similar domains.

Keywords: Model Driven Development, Domain Specific Languages, AI, ML, Health Informatics, XMDD

1 Introduction

Health informatics is a highly multidisciplinary field at the intersection between medicine and information technology. The field seeks to leverage information technology tools to advance all

* This author is sponsored by the Science Foundation Ireland under Grant number 18/CRT/6223

aspects of health care, be it in a clinical or research setting. In recent years health informatics has become more and more data-centric, closely in line with the rapid increase in our ability to gather and store more and different types of data. This has led to an explosion in the use of AI in recent times as we are now able to collect and store repositories of data so large, that they have surpassed our ability to manually extract and develop actionable knowledge from them in an effective or efficient manner. Fortunately, AI algorithms and computational techniques have been developed (and are constantly being developed) that can perform tasks previously only possible by humans to a high degree of accuracy in a small fraction of the time it would take a human, allowing use to be made of all this data.

AI is already used in a broad array of subdomains within the health informatics space. For instance, when it comes to public health, AI techniques are being used for medical surveillance. An AI driven approach enabled researchers to identify food-born illnesses using social media data [HMC+14]. Another study used AI to mine social media data to identify individuals who may be suffering from depression [MSC+16]. Also with respect to public health, AI is being used to assist the public to better assess the quality WBHI as seen in [KAK20], where the authors used AI to automate the evaluation of WBHI with respect to the DISCERN instrument [CSNG99]. In Clinical medicine AI is being used to interpret medical images for diagnosis including; lung cancer screening [AKB+19], breast cancer screening [WPP+19] [MSG+20] and cardiac function assessment [GOA+20]. In biomedical research, AI is being used for a plethora of things such as semantic segmentation of biomedical images [RFB15], the prediction of protein structures [JEP+21] and DNA fragment Assembly [BCC11]. However this is just a brief overview of the use being made of AI in health informatics, the current applications are extremely pervasive and growing rapidly.

The Model-Driven Development (MDD) approach breaks with the code-centric approach to developing software and places models at the centre. Rather than developers spelling out every detail of how a software system works in code, models are used to define the functionality that is needed and the overall requirements for the system's architecture. In the MDD paradigm, models are Platform-Independent Models (PIMs) so that the system's design is not bound to an underlying implementation technology. In essence, it enables the developer to focus on what the system needs to do rather than how it is implemented in code. Domain-Specific Languages (DSLs) are programming languages that offer a higher level of abstraction than general-purpose languages and are optimised for use within a specific problem domain. DSLs encapsulate domain-specific constructs and rules, which make them more comprehensible to domain experts than those in general-purpose languages. The idea of DSLs is not a new one (for example SQL, is a DSL for database operations), but as computing is being applied to an increasing amount and more diverse set of problem domains DSLs provide extensive opportunities for greater productivity due to facilitating greater participation of domain experts in the development process. LCNC development environments are based on the principles of MDD. They are typically comprised of a graphical editor that enables users to develop software in a drag-and-drop fashion to define *what* the system does and define the relationships between models (i.e. data flow). In general LCNC development environments are tailored to a specific domain with the available set of models and possible relationships being a DSL. Through the use of automatic code generation, the

graphical abstractions of the system are translated to executable code or even a production-ready application. Given they require little to no coding experience LCNC development environments facilitate domain experts to actively participate in the development of production-ready applications for use within their domain, either by co-design with a software developer or completely independently.

Whilst AI participation in the health informatics space is growing rapidly, with many research labs in the various sub-domains having collaborators with knowledge of programming in their teams actively developing new AI driven solutions in a variety of health informatics research, there is a hurdle to widespread participation. The vast majority of these tools are being developed in general-purpose programming languages such as Python, meaning there is the prerequisite requirement of programming expertise for domain experts to participate in the benefits of AI in their field. To address this, developing domain-specific LCNC AI platforms would not only facilitate domain experts to use existing AI technologies but build upon them by leveraging their domain knowledge to create new workflows, pipelines and applications. This could ultimately result in greater productivity as they can focus on advancing their research rather than learning to code.

Our goal is to develop tailored LCNC AI platforms for a variety of domains within the Health Informatics space, to ease the technical knowledge burden on those working in those domains. In essence, democratising the use of AI by making it available to those with no expertise in coding. In order to address this goal we chose the CINCO [NLKS18] product DIME [BFK+16] and a bespoke product built in CINCO Cloud [BBK+22] as starting points. Our contribution is extending them by creating DSLs and integrating AI services for two domains; public health and biomedical research. The remainder of the paper is comprised of Section 2 which discusses the related work. Section 3 outlines the principles of the LCNC AI Environments for Health Informatics. Section 4 introduces two case studies that are based on our ongoing projects and finally in section 5 we offer the conclusion and steps to be taken going forward.

2 Related Work

In recent times, textual and Graphical Domain-Specific Languages (GDSLs) following the model-driven approach and the LCNC paradigm have become more and more popular for designing and developing complex systems. According to Gartner's 2022 report on low-code development technologies, the market for low-code technologies is expected to grow by 20% in 2023, with that trend continuing into the future. They also anticipate that 80% of the user base using low-code tools for software development to be people outside formal IT departments, i.e. non-programmers [New22]. This illustrates that LCNC platforms address many of the issues associated with traditional software development paradigms and by lowering the knowledge burden democratize the development of software by enabling domain experts who have little or no expertise in coding to actively participate in the development process.

In the health informatics space, a number of LCNC platforms and DSLs have been devel-

oped in recent times to reduce the complexity of both computational analysis and the use of AI in the field. Galaxy is a software ecosystem first released in 2005 that consists of several software systems that can be accessed via a web browser [JAG+20]. The system available at 'UseGalaxy.org'[1] is tailored toward data-intensive biomedical research and has two features that are pertinent to our work, *Worflow* and *Visualise*. The *Workflow* feature is a no-code browser-based environment that has access to an integrated repository of tools for a wide range of biomedical studies such as sequence and variant analysis, metagenomics, proteomics, and transcriptomics. The *Visualise* feature is a no-code tool that enables exploratory data analysis via the web browser. Whilst Galaxy is a vibrant and well-established ecosystem it doesn't support many uses cases in the health informatics space, two of which are the target of our research. NextFlow [DCF+17] is software for data-driven computational pipelines. It consists of a textual DSL that enables the implementation and deployment of complex parallel and reactive workflows on the cloud or a cluster of computers. It can be deployed on any POSIX-compatible operating system. It provides an abstraction layer between a computational pipelines process logic and the execution layer and therefore is implementation-independent with built-in executors for the likes of SLURM, Kubernetes and several cloud providers. Toil [VRN+17] is an open-source workflow software that can be used to deploy and execute scientific computational workflows at a large scale in the cloud or High Performance Computing (HPC) environments. It is accessed programmatically through an Application Programming Interface (API) in Python. It also supports what the authors describe as a Common Workflow Language (CWL) and Workflow Description Language (WDL), which are essentially two DSLs that separate the process logic from the deployment (similar to NextFlow). While both NextFlow and Toil greatly ease the challenges of deploying computational pipelines they both require coding knowledge and therefore do not achieve the goal our work sets out to achieve. PyCaret is a python-based low-code machine learning library for automating machine learning workflows which can be used for a variety of machine learning tasks [Ali20]. It abstracts away a lot of the code for creating ML workflows, however still requires the users to have a working knowledge of Python programming.

There are several commercial LCNC AI tools available. Microsoft Lobe[2] is a free desktop application that enables users to label data, train a model and then export that model in a variety of formats such as TensorFlow[ABC+16]. It is restricted to Image classification and there is no facility to pick the type of model being trained. h20.ai offer driver-less AI[3], a no-code tool whereby users can build AI models for a variety of tasks using automated techniques developed by h20. Those models can then be deployed via API through their cloud platform or run locally using generated Java code. Amazon Sagemaker[4] has a cloud-based, no-code interface for data preparation and automated AI model training/tuning. The trained model can then be used to make predictions through the web browser. This is not an exhaustive list with plenty of other options available. However, there is a general theme amongst these tools, which is that while in theory they could be used to solve certain problems in the health informatics domain, none of them to the best of our knowledge incorporate the functionality required for creating pipelines

[1] https://usegalaxy.org/
[2] https://www.lobe.ai/
[3] https://h2o.ai/platform/ai-cloud/make/h2o-driverless-ai/
[4] https://aws.amazon.com/sagemaker/studio/

to solve difficult problem domains as is the focus of this work. There are also some highly specialised software systems for AI driven processing biomedical images such as Visiopharm[5] and Halo by Indica Labs[6] which do solve many of the problems our work is addressing, however, these software systems illicit huge license fees which make them inaccessible to the majority of researchers.

MDD is an approach to software development that enables rapid design of flexible and cost-effective applications. Placing the model at the centre of the design process, automatic code generation is leveraged to enable the developer to use drag-n-drop visual interfaces to focus on 'what' the applications should be doing rather than how they are going to implement it in code. Using graphical abstractions, the developer models the system and automatic model-to-code generators are used to transform the models in the form of graphical abstractions to executable code [MCF03]. Following these principles, the Java Application Building Center (jABC) framework [SMN+07] based the design and development of applications on Lightweight Process Coordination and formal models. Leveraging the concept of reusable building blocks, orchestrated into analysable control structures titled Service Logic Graphs, jABC accelerated the development cycle of applications. Extending the MDD paradigm through incorporating domain-specific rules and concepts, tailors the modelling environment towards a specific domain of application. The purpose of this is to increase productivity, reduce development complexity and increase the participation of domain experts who likely have little or no coding expertise. An early example of this in practice for the health informatics domain is Bio-Java Electronic Tool Integration Platform (jETI) [MKS08]. The Bio-jETI Graphical Modelling Environment (GME) enabled domain experts to graphically combine bioinformatics services to create more complex workflows without worrying about details of their interfaces, type mismatches of the composition and most importantly without having to write any code.

The Language-Driven Engineering (LDE) approach [SGNM19], addresses the challenge of adapting the needs of DSLs with the practicalities of minimising the cost, expertise and complexity of developing their corresponding systems from scratch. The CINCO meta-level framework by the same authors as the LDE approach supports a meta-model approach to the development of domain-specific graphical modelling tools that support automatic model-to-code transformation from meta-model specifications. In the LDE paradigm there are two high-level meta-modelling languages, the Meta Graph Language (MGL) and the Meta Style Language (MSL). The MGL describes the language's syntax and semantics, with the MSL describing the rendering style of the language in the graphical editor. Following this LDE approach and using the CINCO framework several MDD tools have been developed for use in a variety of different application domains [ZNS19, ZS21, BFK+16, CKL+16]. With regards to our work we chose to work with two specific products from the Sustainable Computing for Continuous Engineering (SCCE) eco-system, DIME and CINCO Cloud [BBK+22].

DIME is an Eclipse-based, low-code Integrated Modelling Environemnt (IME) for developing

[5] https://visiopharm.com/
[6] https://indicalab.com/halo/

and deploying complete web applications in a service-oriented manner [MSR05]. It follows the principles of the One Thing Approach (OTA) [MS09] through its one-click generation and deployment. Dime supports hierarchical modelling which is useful for designing scalable real-life systems. The modelling language consists of three main model types which are the different aspects of a web application. The data models facilitate the design of data types and persistency. The process models facilitate the design of business logic with respect to control and data flow. Finally, the GUI models facilitate the design of the web applications' user interface. When developing an application in DIME the user modelling is validated dynamically from both a syntactic and static semantics point of view on both a model and project level. This facility supports users to debug their application at the design stage with warnings and error messages helping to localize the issues. This validation at the design stage eliminates the testing overhead present when not following the OTA.

CINCO Cloud is a holistic web-based language engineering environment that covers all aspects of the LDE paradigm. From meta-modelling the DSLs, to modelling an application with those DSLs to deploying the final product via Continuous Integration/Deployment (CI/CD) pipelines with Git repository platform integrations. CINCO Cloud is a fully cloud-based application and therefore does not require the user to install any software locally, with users accessing it via the web browser. CINCO Cloud facilitates real-time collaboration and user access management capabilities. There are three layers in CINCO cloud; the meta layer where the graphical DSLs are designed, the modelling layer where the created DSL is used in the IME to develop an application and the product layer where the deployment of the developed application is handled through the integrations with Git repository platforms. Similar to DIME, both the meta and modelling layers in CINCO cloud provide syntax validation to aid the user in debugging their application at design time.

LCNC development environments and DSLs promise to fulfil robust application requirements, automate the majority of the software development life-cycle and address several challenges that are associated with conventional software development. As has been shown by many of the developments in other domains, domain-specific LCNC development environments have massive potential to democratize the use of AI driven computational analysis in the health informatics domain. Given this potential they are likely to be an extremely fruitful avenue to increase the productivity of and ease the technical knowledge burden on domain experts, leaving them to focus on their actual research.

3 Developing LCNC AI Environments for Health Informatics

Our work endeavours to create a number of AI platforms for a variety of subdomains in health informatics by extending the AI service functionality of the base platforms DIME and CINCO cloud through integrating existing AI technologies and also ones we have developed to solve problems in our chosen application domains. The target of the work initially focuses on informatics for public health and biomedical research. Specifically offering a LCNC AI driven web application development environment in DIME, to aid researchers, public organisations and charities to

build applications that leverage AI to benefit public health interests. With respect to biomedical research, we offer a platform to model and execute AI driven workflows for the processing and computational analysis of biomedical images, initially targeted at tissue-based high dimensional spatial proteomics.

3.1 The DIME platform architecture

The current architecture of the IME for the public health AI web application platform is based on DIME. DIME is an IME for developing and deploying full-stack web applications. The DIME IME breaks down the development of a web application into three distinct model types; Graphical User Interface (GUI) models, process models and data models, also supporting hierarchical modelling. GUI models are used to model the User Interface (UI) for an application and it follows the principle of What You See Is What You Get (WYSIWYG), this enables rapid prototyping of UIs. Data models are used to define the data types and persistence for an application and the relations between different data types. Process models are used to model the business logic, both control flow and data flow, for an application, with each process model in essence being a compartmentalised piece of logic that when pieced together with other process models defines the overall business logic for an application. DIME has Common DSLs which are native to the default distribution and support a number of generic web application requirements such as manipulation of strings, mathematical operations, etc.

In the case of domain-specific requirements not natively supported in DIME it is possible to extend DIME with new External native DSLs. External Native DSLs are a set of external services that exist outside the DIME environment that can be accessed at runtime via network protocols, thereby integrating functionalities that are tailored to a specific domain that can be modelled in DIME. The Process DSL is the language definition that handles the orchestration of domain-specific, generic and hierarchical workflows of Common and External Native DSLs. The focus of our work with respect to DIME is developing new external services in a MDD and service-orientated fashion, then integrating those services into DIME as External Native DSLs and developing Process DSLs which handle the orchestration of those services to enable them to be used easily by domain experts when building their applications.
We found that the majority of the functionality for supporting our target use case was not natively supported in DIME. Therefore DIME needed to be extended with a new set of DSLs which met the requirements for designing such a web application. Whilst there are Java (the native language for DIME) libraries available for Natural Language Processing (NLP) and Machine Learning (ML) such as Apache Spark MLib[7], the options are somewhat limited. Python, on the other hand, has an extremely vibrant ecosystem with NLP and AI libraries [PVG+11, LB02, WDS+19, PGM+19, ABC+16] and so was the preferred option for implementing the functionality. Given that Python requires a different runtime infrastructure than a native DIME application, the service integration methodology from prior work [CM21] was followed. Once we had decided on the most suitable programming language for the services to be implemented in and a means of integrating those services into DIME, all that remained was implementing the services

[7] https://spark.apache.org/mllib/

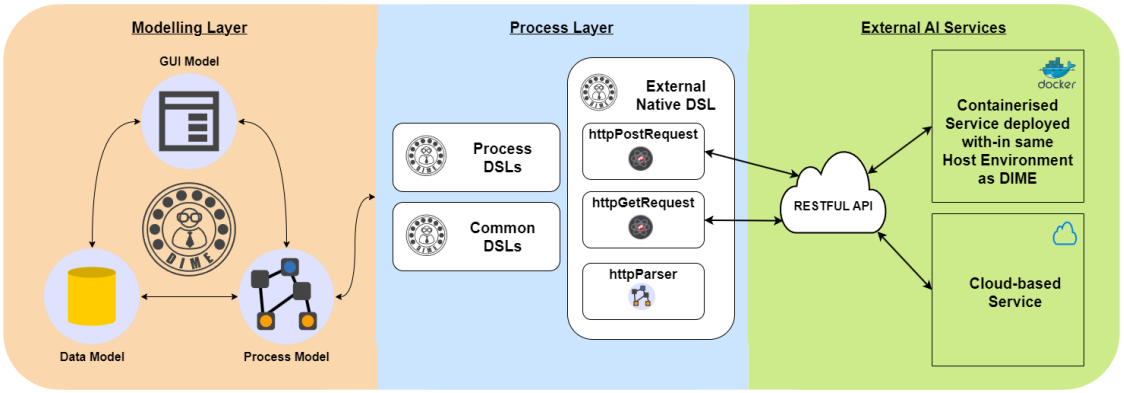

Figure 1: Overview of the DIME architecture for the LCNC AI Web App development platform

in code and deploying them. Before any service implementation code was written, the first step was to design the models for the various AI processing pipelines that comprise a system of this nature. This was done using the blueprinting facility in DIME, with each processing pipeline being broken down into a set of atomic processing steps and a model of the inputs and outputs for each processing step being blueprinted. Given that all the domain-specific functions are implemented as external services, this approach greatly aided in choosing the level of abstraction that becomes a Service-Independent Building Block (SIB) and which becomes a Process DSL, and therefore how the services were implemented in code. Figure 2 shows the blueprint of a NLP pipeline for taking raw text in the form of a string, applying a number of pre-processing steps and inputting it into some classifier to get some label. Following the blueprinting approach aided in making the decision to be in-keeping with the premise of SIBs, reusability and generability of operation were central to the decisions. For example, deciding that **tokenization** was implemented and deployed as a standalone service, then integrated into DIME as a SIB rather than the entire pipeline seen in Figure 2 being implemented, deployed and integrated as a SIB. The pipeline would instead form a Process DSL. The set of SIBs for 'tokenization', 'token_cleaning' and 'stop_word_removal' as denoted in Figure 2 collectively would form the basis of a NLP pre-processing DSL.
The implementation code for the services was then written in Python for the chosen set of models. The containerised service-orientated approach was chosen due to the heterogeneous landscape of technologies, platforms and dependencies for the various AI ecosystems. When deploying these services two avenues were taken; local deployment and cloud deployment. For local deployment the services and their dependencies were containerised, made accessible via a REST APIs and deployed in the same host environment as the deployed DIME application. For cloud deployment, the services and their dependencies were deployed in AWS Lambda and made accessible through a REST API via API Gateway.

3.2 The CINCO Cloud Architecture Extension

The current architecture of the AI driven computational image processing platform has as its base CINCO Cloud the web-based language engineering environment. A high-level overview of the

Figure 2: A generic raw text to classification NLP pipeline blueprinted in DIME

standard distribution of Cinco Cloud architecture can be seen in Figure. 3. In the typical use case for CINCO cloud the MSL, MGL and CINCO Product Definition (CPD) are used within meta-IME (meta layer in Figure. 3) to define an Application Domain Specific Language (A-DSL) and IME for the target use case. Accompanying this IME, code-generators are written which will translate the models into executable source code. This is what is referred to as a CINCO product. In the original desktop application of CINCO, the generated CINCO product was a standalone desktop application (such as DIME), however in CINCO cloud the CINCO product simply becomes a new project that can be opened in the same interface as the Meta-IME. With-in the CINCO product, a Modelling Domain Specific Language (M-DSL) can be created through the creation of SIBs (like the extensions of DIME in 3.1) which then are used to model the functionality of an application within the structure provided by the A-DSL. Users can then define their application in the IME and subsequently generate the application code with the model-to-code transformer. The actual user-developed application referred to as the *"product"*

exists in what the CINCO developers call the product layer and the generated source code is then pushed to an external repository (such as GitLab) and can subsequently be deployed by the user where appropriate. The requirements for the target use case deviate from what is supported

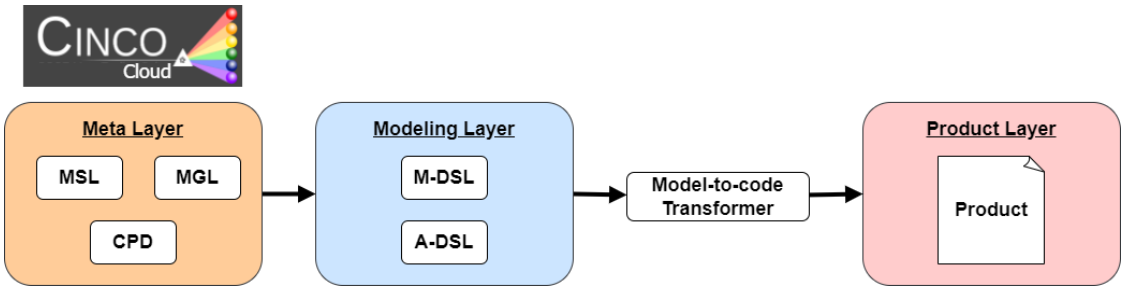

Figure 3: Standard Cinco Cloud distribution architecture

by the standard CINCO Cloud distribution. In our use case, the user will not be modelling a standalone application that is deployed externally, but rather a script that orchestrates a set of existing services that then needs to be executed somewhere. We are calling this extension to the standard CINCO Cloud distribution, CINCO de Bio and it consists of four modular components we refer to as layers; the meta layer, the modelling layer, the execution layer and the service layer. The modelling, execution and service layers communicate with each other via network protocols. Like in the standard CINCO Cloud development process, we make use of the Meta Layer to develop a M-DSL that captures the syntactic and semantic requirements for orchestrating an AI driven image processing workflow. However, along with creating this CINCO product the standard CINCO Cloud deployment needed to be extended with an execution layer that would take the generated workflow code and execute it and the service layer which contains the services that process the data. The A-DSL is comprised of a collection of SIBs that represent the set of integrated AI and other computational processing services that have been integrated into the service layer. We are currently developing a system for automatically passing model definitions from the service layer to the modelling layer, so as soon a service is integrated into the service layer, an accompanying SIB will be added to the A-DSL in the modelling layer.
As processing these images is very data-intensive, the design decision was taken to pass symbolic links of the processed data from the services to the execution environment. The actual data is stored in an object storage system which exists within the services layer. With respect to implementing and deploying the AI services, a nearly identical approach was followed as in 3.1, with the services first modelled, manually transformed to code, and made accessible via Representational State Transfer (REST) APIs and containerised.

4 Case Studies

Integration of AI in the public health domain is extremely important for improving the health outcomes for society, specifically when it comes to citizens' ability to access high-quality health information on the internet. With respect to biomedical research, the integration of AI has the potential to revolutionise the analysis of the complex processes that occur in the human body.

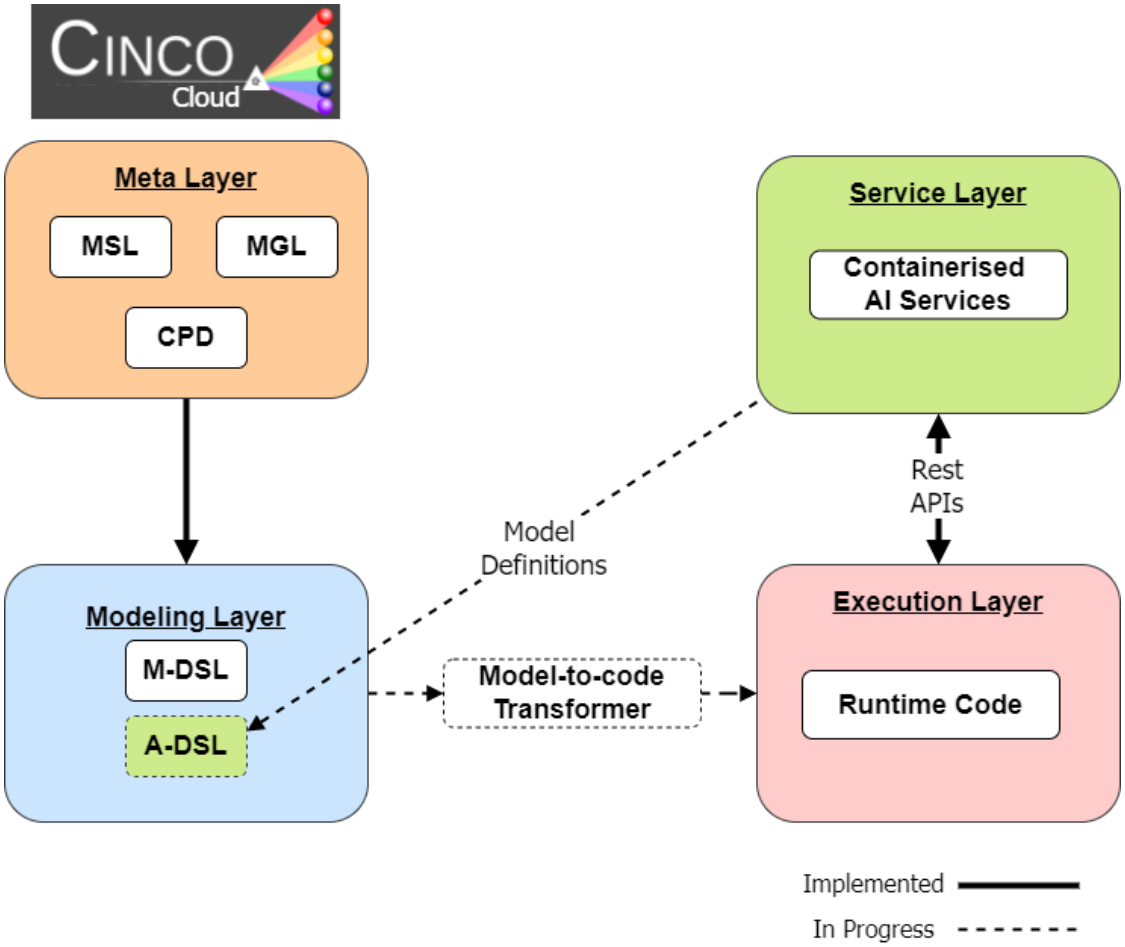

Figure 4: A high-level overview of the extension to the CINCO Cloud architecture

Furthering our understanding of complex diseases, facilitates the development of highly targeted medical/pharmaceutical interventions and beyond. We discuss in 4.1 and 4.2 how we implemented the required functionality in the DIME and CINCO Cloud platforms to build AI LCNC platforms for use in the public health and biomedical research domains.

4.1 Health Information Pipeline for Patients and the Public (HIPPP)

The HIPPP case study aims to enable public health domain experts or physicians to co-develop an AI driven pipeline for automatically evaluating the quality of WBHI from a user-provided URL that pertains to colon cancer, using the AI LCNC Platform we created through extending DIME. This work is being done in collaboration with the UL Cancer Network (ULCan). The evaluation system is based on the QUality Evaluation Scoring Tool (QUEST) framework [RJLF18], a manual framework developed and validated by physicians for quantitatively assessing the quality of WBHI. In essence, the goal was to use an ensemble of AI techniques and other computational

processes to automate the assessment of a piece of WBHI with respect to the framework.

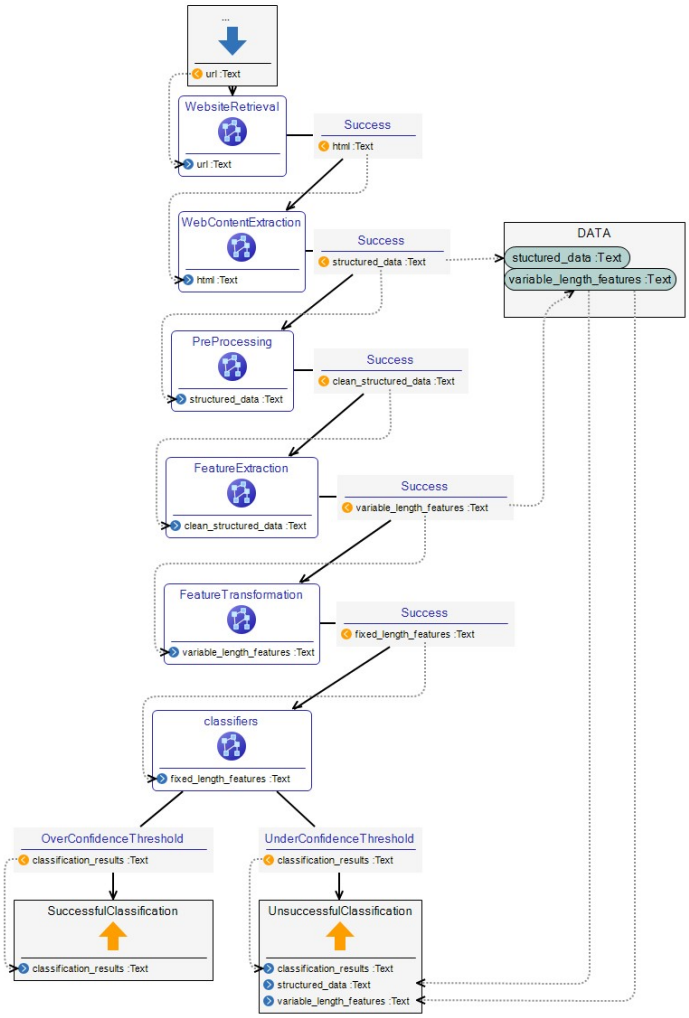

Figure 5: Top-level process model of the HIPPP system

Figure 5 provides the top-level overview of the control and data flow for the HIPPP system. The control flow process from the Start (blue arrow) with a user-provided URL. The user-provided URL is passed into the ”websiteRetrieval” SIB, which makes an HTTP request to the provided URL, parses the response and gets the content which is an HyperText Markup Language (HTML) document as a String. This HTML passes into the ”webContentExtraction” SIB, which takes the HTML document (unstructured or semi-structured data) and uses AI models to extract relevant data (such as article body, author, date of publication, etc.) and store it in a structured format as a JSON string. This structured data is passed into the ”preProcessing” SIB which applies natural language pre-processing (tokenisation, coreference resolution, etc.) and network pre-processing

(conversion to graph representations) to optimally prepare the data for being input to the AI models used in downstream processes, again the pre-processed data is stored in a structured format as a Javascript Object Notation (JSON) string. This now cleaned and pre-processed data is passed into the ”featureExtraction” SIB which uses a variety of AI models to extract features from the data which facilitates classification with respect to the QUEST framework. We manually designed the set of features to be extracted so as to leverage as many existing AI algorithms as possible. This step uses the chosen AI models to extract these numerical feature vectors which are then stored along with their confidence values in a structured format as a JSON string. The data is now passed into the ”featureTransformation” SIB. Owing to the fact that WBHI can be of different lengths, heterogeneous formats, etc. the feature vectors extracted in the previous step are of variable length. Classifiers (both AI and rule-based) require input feature vectors of a predefined length, the feature transformation process uses statistical methods, dimensionality reduction and neural networks to create fixed-length representations. As with the previous steps they are stored in a structured format as a JSON string. The fixed length features then pass into the classifiers SIB, which is comprised of an ensemble of classifiers with each classifier pertaining to a specific evaluation criterion (authorship, attribution, type of study, conflicts of interest, currency, complementary, tone) from the QUEST framework. Each classifier assigns a QUEST score for that criteria based on the relevant set of features. Finally, a confidence analysis is done based on the confidence of the feature extraction models, if the aggregate confidence across the entire evaluation exceeds a predefined threshold, then the evaluation results are presented to the user. In the case, it does not meet the threshold the evaluation is referred to an expert to be manually reviewed.

Figure 6 shows a more granular view of a portion of the overall pipeline shown in Figure 5. It can be seen that the *”GetWebsite”*. and *”WebContentExtraction”* SIBs, are both self-contained complex processes made up of a number of atomic SIBs, i.e. *URL validation* and *soft404 [PÁC14] detection* in the case of *”GetWebsite”*.

Figure 7 provides an example of how the system for automating the QUEST evaluation framework was modelled (this is for illustrative purposes as in the actual pipeline these functions are compartmentalised and displaying the hierarchical processes would be unwieldy). To the left of the Figure, an excerpt of the QUEST framework pertaining to evaluating a piece of WBHI with respect to authorship is shown. To the right, a simplified blueprint of the DIME process model as to how that was achieved is shown. The HTML document passes into the ”webContentExtraction” SIB which outputs the author element in the form of a string. This string then passes into the ’preprocessAuthorElement” SIB, which prepares the data for the AI models. There is a branch at the output of this SIB, as if there is no text in the author element, that automatically denotes that there is ’no indication of authorship or username” and so the analysis ends, with a score of 0 being assigned. If the string is not empty. the tokens and part-of-speech tags, are passed into the ”namedEntityRecognition” SIB, this contains an AI model which performs Named Entity Recognition (NER) [NS07], which identifies if there are any named entities in the sequence of text. The named entities are output as a list of Strings and pass into the ’personFound’ SIB. This is a simple rule-based classifier that checks if there is an entity of type Person in the list of named entities. There are two branches which can be taken at this point, if no person is found, it

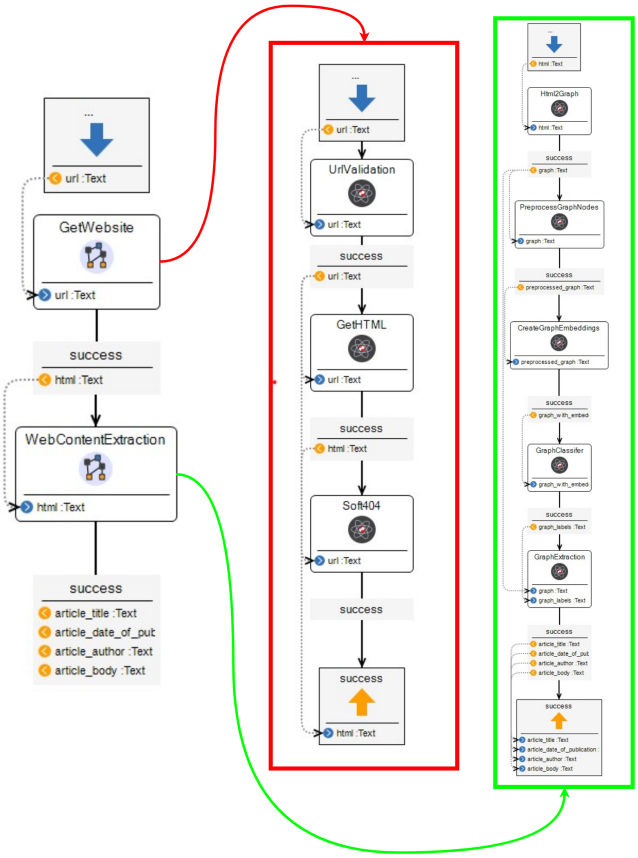

Figure 6: Hierarchical process SIBs for **Website Retrieval** and **Web Content Extraction**.

follows the 'False' branch and again as this denotes there is no indication of authorship a score of 0 is assigned. If the 'True' branch is followed, the list of named entities is passed into the 'professionFound' SIB, similar to the 'personFound' SIB, this SIB simply checks in the list of named entities if there is an entity of type Profession (i.e. job title). Again there are two branches that can be taken here depending on the outcome, if 'False' then this corresponds to 'All other indications of authorship' as the person has already been found and a score of 1 is assigned. If 'True' then this denotes that the 'Author's name and qualification clearly stated' and therefore a score of 2 is assigned.

We have recently received ethical approval to run Patient and Public Involvement (PPI) groups which will be used for validating the data collection techniques we used to train the variety of AI models developed for the pipeline. Once we have finished that process, we will deploy the HIPPP web application and make it available to the public and incrementally refine the system over time. The majority of SIBs, DSLs and processes we have developed to date for this case study are reusable across other use cases and domains. In essence, we have created a library of DSLs in DIME which could be used for any form of natural language or network-based analysis of WBHI including building a similar web application to HIPPP.

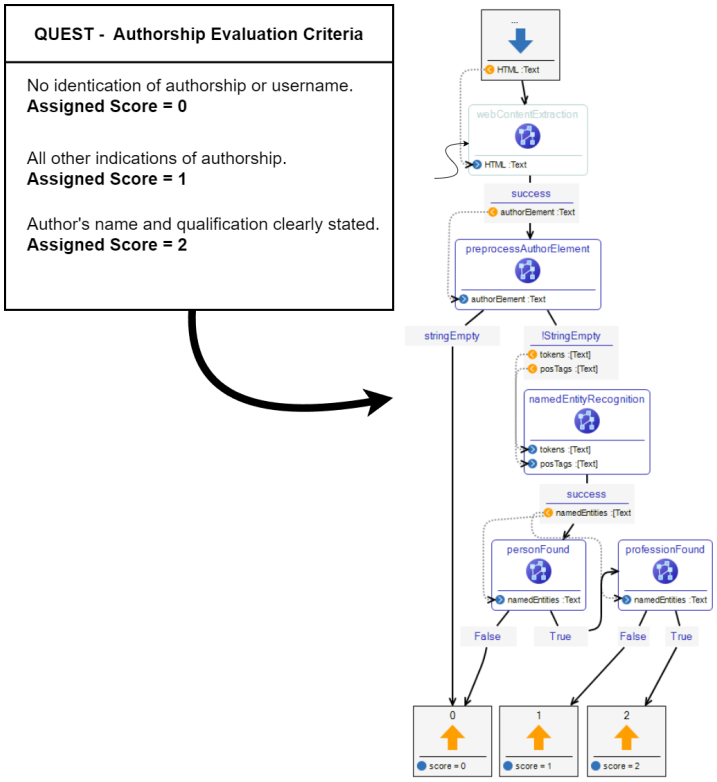

Figure 7: An example in DIME of how the QUEST authorship evaluation criteria were translated to AI and computational methods in order to automate the evaluation.

4.2 Workflow Environment for Computational Analysis of Highly-plexed Tissue-Images (WE CAHT)

The WE CAHT case study aims to facilitate biomedical researchers to build, deploy and execute bespoke image processing workflows using a bespoke tool that extends CINCO Cloud, which is under development as reported in Figure 4. This is a collaboration with the SCCE groups in TU Dortmund and University Limerick and the Bernal Institute also in the University of Limerick. An example of a basic workflow orchestration is seen in Figure 9: the user inputs a highly-plexed slide image obtained from their experiment with the final output being a Comma Seperated Value (CSV) file which contains the protein signal and morphological information for each cell in their sample.

Image Splitting The experimental imaging machines output a slide image in the format of a stacked Tagged Image File Format (TIFF) file. These images are extremely high resolution (20000 x 20000 pixels) and are highly-plexed (have at least 40 colour channels). It also outputs a text file that contains the list of protein channels imaged by the machine and their index in the stacked TIFF file. Image splitting takes the original image matrix with dimensions W x H x N and transforms it to a set of size N, n in N being a grey-scale image

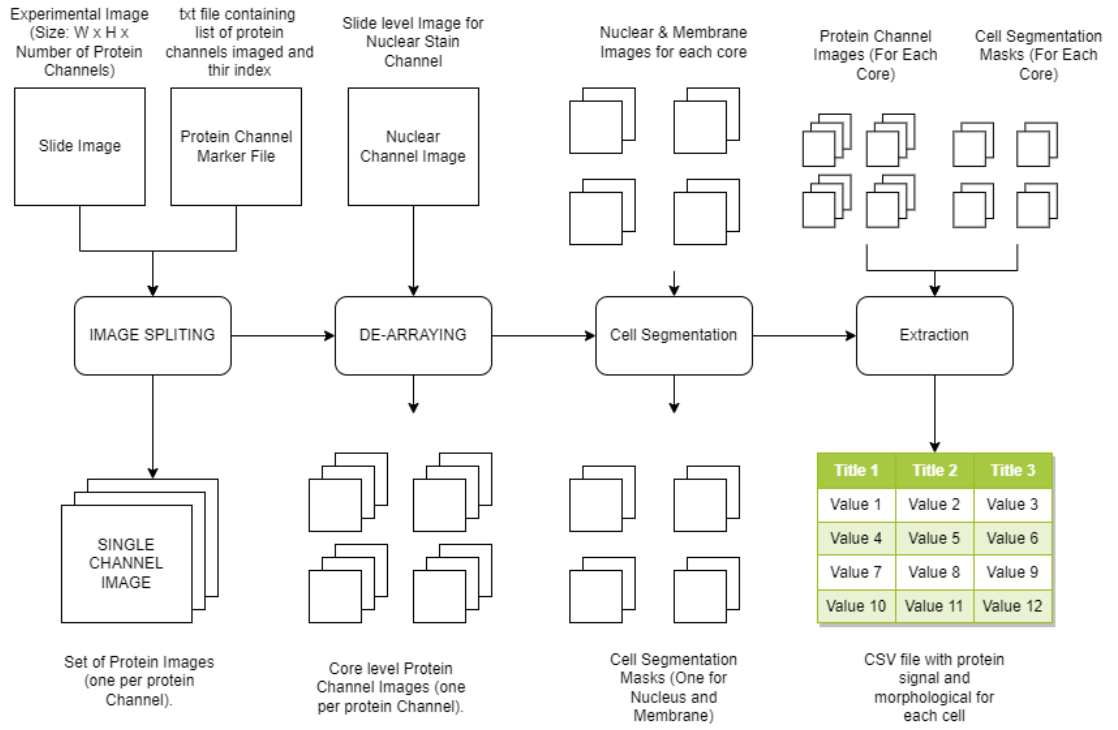

Figure 8: Overview of a basic highly-plexed image processing workflow.

with dimensions W x H x 1 as this is the required format for subsequent processing steps.

Image Splitting The slide image typically contains several tissue samples (called cores), each core needs to be analysed separately as they're either from different people or they're from different tissue types. De-Arraying is the process of semantic segmentation on the slide image, to find the bounding boxes of individual cores, the semantic segmentation is performed on a single channel image, the nuclear stain channel Image as denoted in the channel markers file (the most common stain is 4',6-diamidino-2-phenylindole or DAPI). Once the semantic segmentation finds the segmentation masks, the bounding boxes for the respective cores are found. The regions enclosed by those bounding boxes then need to be cropped out for each protein channel image. This results in a set of sets of images, with the outer dimension being the number of cores and the inner dimension being the number of protein channels.

Cell Segmentation For each of the cores, cell segmentation finds the nuclei and membrane within those tissue samples. Taking the nuclear channel marker and the membrane channel marker (these are the protein channels, that a biologist knows best denote the nuclei and membranes of a cell respectively) the cell segmentation predicts the segmentation masks of the nucleus and membrane in each cell.

Extraction The actual analysis (clustering, statistical tests, etc..) that a biologist will seek to do is on cellular data in a tabular format. The extraction step takes in the cell segmentation

masks (nucleus and membrane) for each core and its set of protein channel images. It then finds the level of signal for each protein (denoted by pixel intensity in a channel image) within the nucleus and membrane regions for each cell. It also extracts morphological features for each cell based on the segmentation masks.

Following similar principles to those in the HIPPP case study, the system requirements were modelled and then transformed to code. Taking as an example the process to de-array a slide-level image, the nuclear stain image first has to be transformed into a Tensor (the expected format for the model(s)). The segmentation model then takes the image in tensor format and makes a prediction, the output is in the format of a probability map (a tensor of floats between 0 and 1). Gating is applied to this probability map using a tuned confidence threshold which yields a binary Tensor, this is then converted to an Image. A combination of edge detection, bounding rectangle detection and non-maximum suppression is then applied to find the regions of interest that contain each of the objects of interest. The image split-er takes in the regions of interest and uses them as the parameters for cropping every slide-level image for each protein channel. A high level of abstraction was chosen for the DSL after consultation with domain experts as users of this DSL would likely find it difficult to build their own de-arraying pipelines as they would need pre-requisite knowledge of computer vision techniques. We elected to build a set of de-arraying workflows that incorporate different models and techniques. Domain users can then choose the de-array SIB that best fits their needs. The same principles were applied to the other workflow tasks. The services layer is implemented with the required services and their

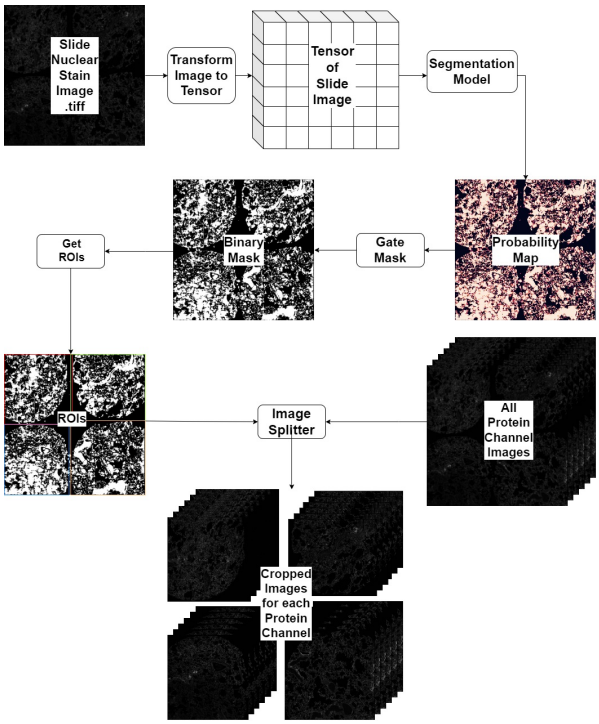

Figure 9: Overview of the modelled process for Slide Image De-arraying

respective model definitions, however, we are incrementally adding additional AI models and other functionalities. With respect to the integration between the services layer and the modelling, work is ongoing from our collaborators in TU Dortmund to develop the system that pulls the model definitions from the services layer in as SIBs to populate the application DSL in the IME. Work is also ongoing on the model-to-code transformer which will generate the run-time code that executes the workflow in the execution layer. An execution layer has been developed for testing, which facilitates the services layer to be used as a low-code DSL in Python.
Once the work is complete, the services (represented as SIBs) can be used within the CINCO Cloud-based IME in a no-code drag-and-drop as shown in Figure 10 . Once the user has con-

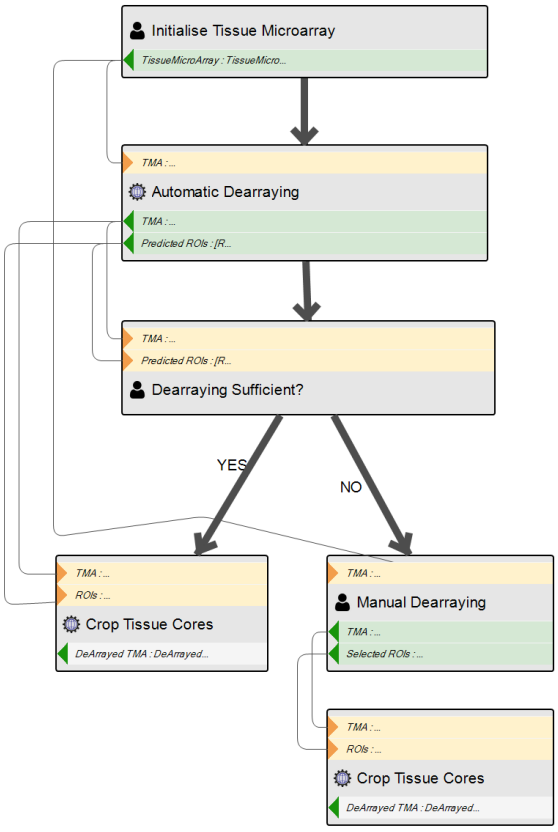

Figure 10: A simple de-arraying workflow modelled in prototype of the IME for the WE CAHT platform

nected to components with the correct data flow the workflow can then be executed. Data is modelled using semantic typing tailored to the domain, i.e. ”Core Segmentation Mask” and ”Nucleus Segmentation Mask”, rather than file extensions or types from the programming language to make it more familiar to domain experts. The WE CAHT platform, will, therefore, enable domain experts the facility to use AI and other computational functionality to process

these images without any coding knowledge or expertise.

5 Conclusion and Next Steps

What was presented in this paper are the core principles, the architectures and individual components of the LCNC AI development platforms for health informatics that we are incrementally developing. We presented two case studies, one that pertains to developing web applications for AI driven WBHI evaluation, with the other pertaining to developing and orchestrating AI driven computational pipelines for processing biomedical images.

Going forward we seek to further refine the current iterations of the architectures to make performance improvements. The integration of additional services to both platforms and development of new AI driven tools to solve a variety of issues brought to our attention by collaborators. We also seek to extend the platforms further so that domain experts can train, fine-tune and deploy their own AI models. This is an important feature due to the requirement of approval from many stakeholders and other constraints, which lead to data sets that contain health or biological data are not made public or shared with other researchers, resulting in data scarcity issues for AI experts to create models. For example, when it comes to tissue data, some labs may work with lung tissue and if for example a publicly available model was trained on tonsil tissue it won't perform as well on lung tissue. For the lab with the lung tissue to share its data with an external AI expert would require many layers of ethical approval making it a slow and unwieldy process. Therefore adding the capacity for a non-AI expert to label data and use it to fine-tune a model or train a model from scratch, then subsequently deploying it as a part of their application or workflow would be of great value to domain researchers.

Acknowledgements: This work was conducted with the financial support of the Science Foundation Ireland Centre for Research Training in Artificial Intelligence under Grant No. 18/CRT/6223.

Bibliography

[ABC+16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al. Tensorflow: a system for large-scale machine learning. In *Osdi*. Volume 16(2016), pp. 265–283. 2016.

[AKB+19] D. Ardila, A. P. Kiraly, S. Bharadwaj, B. Choi, J. J. Reicher, L. Peng, D. Tse, M. Etemadi, W. Ye, G. Corrado et al. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine* 25(6):954–961, 2019.

[Ali20] M. Ali. PyCaret: An open source, low-code machine learning library in Python. April 2020. PyCaret version 1.0.0.
https://www.pycaret.org

[BBK+22] A. Bainczyk, D. Busch, M. Krumrey, D. S. Mitwalli, J. Schürmann, J. Tagoukeng Dongmo, B. Steffen. CINCO cloud: a holistic approach for web-based language-driven engineering. In *Leveraging Applications of Formal Methods, Verification and Validation. Software Engineering: 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22–30, 2022, Proceedings, Part II*. Pp. 407–425. 2022.

[BCC11] M.-I. Bocicor, G. Czibula, I.-G. Czibula. A reinforcement learning approach for solving the fragment assembly problem. In *2011 13th international symposium on symbolic and numeric algorithms for scientific computing*. Pp. 191–198. 2011.

[BFK+16] S. Boßelmann, M. Frohme, D. Kopetzki, M. Lybecait, S. Naujokat, J. Neubauer, D. Wirkner, P. Zweihoff, B. Steffen. DIME: a programming-less modeling environment for web applications. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications: 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II 7*. Pp. 809–832. 2016.

[CKL+16] M. Chadli, J. H. Kim, A. Legay, L.-M. Traonouez, S. Naujokat, B. Steffen, K. G. Larsen. A model-based framework for the specification and analysis of hierarchical scheduling systems. In *Critical Systems: Formal Methods and Automated Verification: Joint 21st International Workshop on Formal Methods for Industrial Critical Systems and 16th International Workshop on Automated Verification of Critical Systems, FMICS-AVoCS 2016, Pisa, Italy, September 26-28, 2016, Proceedings 21*. Pp. 133–141. 2016.

[CM21] H. A. A. Chaudhary, T. Margaria. Integrating external services in DIME. In *Leveraging Applications of Formal Methods, Verification and Validation: 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021, Rhodes, Greece, October 17–29, 2021, Proceedings 10*. Pp. 41–54. 2021.

[CSNG99] D. Charnock, S. Shepperd, G. Needham, R. Gann. DISCERN: an instrument for judging the quality of written consumer health information on treatment choices. *Journal of Epidemiology & Community Health* 53(2):105–111, 1999.

[DCF+17] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, C. Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology* 35(4):316–319, 2017.

[GOA+20] A. Ghorbani, D. Ouyang, A. Abid, B. He, J. H. Chen, R. A. Harrington, D. H. Liang, E. A. Ashley, J. Y. Zou. Deep learning interpretation of echocardiograms. *NPJ digital medicine* 3(1):10, 2020.

[HMC+14] J. K. Harris, R. Mansour, B. Choucair, J. Olson, C. Nissen, J. Bhatt. Health department use of social media to identify foodborne illness—Chicago, Illinois, 2013–2014. *Morbidity and Mortality Weekly Report* 63(32):681, 2014.

[JAG+20] V. Jalili, E. Afgan, Q. Gu, D. Clements, D. Blankenberg, J. Goecks, J. Taylor, A. Nekrutenko. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2020 update. *Nucleic acids research* 48(W1):W395–W402, 2020.

[JEP+21] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko et al. Highly accurate protein structure prediction with AlphaFold. *Nature* 596(7873):583–589, 2021.

[KAK20] L. Kinkead, A. Allam, M. Krauthammer. AutoDiscern: rating the quality of online health information with hierarchical encoder attention-based neural networks. *BMC medical informatics and decision making* 20(1):1–13, 2020.

[LB02] E. Loper, S. Bird. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[MCF03] S. J. Mellor, T. Clark, T. Futagami. Model-driven development: guest editors' introduction. IEEE Software, 20 (5). pp. 14-18. ISSN 0740-7459. *IEEE software* 20(5):14–18, 2003.

[MKS08] T. Margaria, C. Kubczak, B. Steffen. Bio-jETI: a service integration, design, and provisioning platform for orchestrated bioinformatics processes. *BMC bioinformatics* 9(4):1–17, 2008.

[MS09] T. Margaria, B. Steffen. Business process modeling in the jABC: the one-thing approach. In *Handbook of research on business process modeling*. Pp. 1–26. IGI Global, 2009.

[MSC+16] D. Mowery, H. A. Smith, T. Cheney, C. Bryan, M. Conway. Identifying depression-related tweets from Twitter for public health monitoring. *Online Journal of Public Health Informatics* 8(1), 2016.

[MSG+20] S. M. McKinney, M. Sieniek, V. Godbole, J. Godwin, N. Antropova, H. Ashrafian, T. Back, M. Chesus, G. S. Corrado, A. Darzi et al. International evaluation of an AI system for breast cancer screening. *Nature* 577(7788):89–94, 2020.

[MSR05] T. Margaria, B. Steffen, M. Reitenspieß. Service-oriented design: the roots. In *Service-Oriented Computing-ICSOC 2005: Third International Conference, Amsterdam, The Netherlands, December 12-15, 2005. Proceedings 3*. Pp. 450–464. 2005.

[New22] G. Newsroom. Gartner forecasts worldwide low-code development technologies market to grow 20% in 2023. Dec 2022.
https://www.gartner.com/en/newsroom/press-releases/
2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-perc

[NLKS18] S. Naujokat, M. Lybecait, D. Kopetzki, B. Steffen. CINCO: a simplicity-driven approach to full generation of domain-specific graphical modeling tools. *International Journal on Software Tools for Technology Transfer* 20:327–354, 2018.

[NS07] D. Nadeau, S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26, 2007.

[PÁC14] V. M. Prieto, M. Álvarez, F. Cacheda. Soft-404 Pages, A Crawling Problem. *J. Digit. Inf. Manag.* 12(2):73–92, 2014.

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32, 2019.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12:2825–2830, 2011.

[RFB15] O. Ronneberger, P. Fischer, T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Pp. 234–241. 2015.

[RJLF18] J. M. Robillard, J. H. Jun, J.-A. Lai, T. L. Feng. The QUEST for quality online health information: validation of a short quantitative tool. *BMC medical informatics and decision making* 18:1–15, 2018.

[SGNM19] B. Steffen, F. Gossen, S. Naujokat, T. Margaria. Language-driven engineering: from general-purpose to purpose-specific languages. *Computing and Software Science: State of the Art and Perspectives*, pp. 311–344, 2019.

[SMN+07] B. Steffen, T. Margaria, R. Nagel, S. Jörges, C. Kubczak. Model-driven development with the jABC. In *Hardware and Software, Verification and Testing: Second International Haifa Verification Conference, HVC 2006, Haifa, Israel, October 23-26, 2006. Revised Selected Papers 2*. Pp. 92–108. 2007.

[VRN+17] J. Vivian, A. A. Rao, F. A. Nothaft, C. Ketchum, J. Armstrong, A. Novak, J. Pfeil, J. Narkizian, A. D. Deran, A. Musselman-Brown et al. Toil enables reproducible, open source, big biomedical data analyses. *Nature biotechnology* 35(4):314–316, 2017.

[WDS+19] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[WPP+19] N. Wu, J. Phang, J. Park, Y. Shen, Z. Huang, M. Zorin, S. Jastrzebski, T. Févry, J. Katsnelson, E. Kim et al. Deep neural networks improve radiologists' performance in breast cancer screening. *IEEE transactions on medical imaging* 39(4):1184–1194, 2019.

[ZNS19] P. Zweihoff, S. Naujokat, B. Steffen. Pyro: generating domain-specific collaborative online modeling environments. In *Fundamental Approaches to Software Engineering: 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings 22*. Pp. 101–115. 2019.

[ZS21] P. Zweihoff, B. Steffen. Pyrus: an online modeling environment for no-code data-analytics service composition. In *Leveraging Applications of Formal Methods, Verification and Validation: 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021, Rhodes, Greece, October 17–29, 2021, Proceedings 10*. Pp. 18–40. 2021.