Chapter in Knowledge Management in Development of Data-intensive Software Systems, CRC Press/Taylor & Francis, Dec 2020

Knowledge Management via Human-centric, Domain-Specific Visual Languages for Data-intensive Software Systems

John Grundy[*], Hourieh Khalajzadeh[*], Andrew J. Simmons[†], Humphrey Obie[*], Mohamed Abdelrazek[†], John Hosking[‡], Qiang He[§]

[*]*Faculty of Information Technology, Monash University*, Melbourne, Australia
{john.grundy, hourieh.khalajzadeh, humphrey.obie}@monash.edu

[†]*Faculty of Science, Engineering, and Built Environment, Deakin University*, Melbourne, Australia
{a.simmons, mohamed.abdelrazek}@deakin.edu.au

[‡]*Faculty of Science, University of Auckland*, Auckland, New Zealand
j.hosking@auckland.ac.nz

[§]*School of Software and Electrical Engineering, Swinburne University*, Melbourne, Australia
qhe@swin.edu.au

Abstract—We describe the use of a suite of human-centric, domain-specific visual languages to manage knowledge for data-intensive systems. We use two exemplar system case studies – a smart home to support ageing people and a set of smart city technologies – to motivate the need for such an approach. We then describe aspects of these two example systems from abstract requirements to specific data analysis, implementation and deployment choices using our BiDaML representation. We discuss the strengths and limitations of the approach, and key directions for further work in this area.

Index Terms—data-intensive systems; knowledge management smart homes; smart cities; internet of things; human factors; domain-specific visual languages

I. Introduction

Smart living applications are increasingly in demand. These range from smart homes – for general-purpose use or specific tasks e.g. supporting aged care, disability or rehabilitation; smart transport systems; smart grid and other utility-oriented systems; autonomous vehicles; 'smart living' robotics; industry 4.0-supporting applications; smart hospitals and schools; and smart buildings in general [1]–[3].

For example, Figure 1 shows two exemplar smart living solutions, a smart home to support ageing people living in their homes, and a smart urban environment. Both are composed of a number of sensors and interactors. Both have a range of diverse end users with diverse end user requirements. The sensors generate significant amounts of heterogeneous data that needs to be stored, processed, and presented. Some of the data processing is done "on the edge" [4]. Other is uploaded and done "on the cloud" [5]. These lead to a need for careful algorithm selection for the data processing, choice of deployment of algorithms, and management of security, privacy, reliability, performance and robustness constraints.

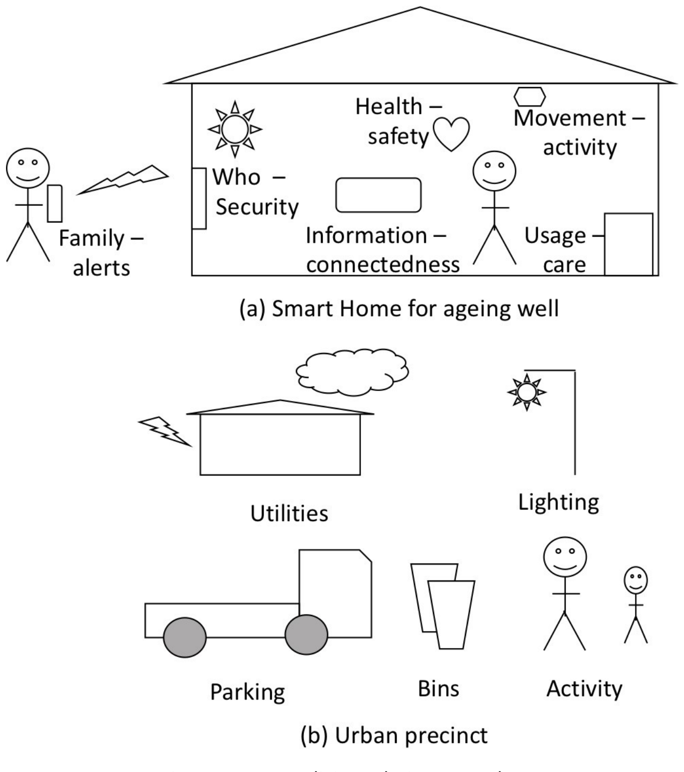


Fig. 1. Two smart living solution examples

Such systems come with a number of serious challenges in the engineering of the system. By their very nature, they need to incorporate a wide range of users and use cases, leading to challenging requirements engineering tasks. This includes determining the high-level requirements of the system, key processes it needs to support, key data sources, data storage and data processing, and required system and user interactions [6]. Architecting such systems is challenging [7], given the wide range of hardware and application software platforms needed. Many design decisions relate to the data-intensive nature of the system: how to obtain, wrangle, transform, integrate, harmonise and store diverse data; choice of machine learning techniques to process and extract key information from the data; and appropriate large scale data visualisation techniques needed to support diverse decision-making.

We have developed a suite of domain-specific visual languages to assist in supporting the development of such data-intensive systems, BiDaML [8]. BiDaML provides a set of visual languages to capture high level to detailed requirements, data-oriented design decisions, deployment scenarios, and detailed data representation and processing formats and algorithms. In this chapter, we illustrate the use of our BiDaML approach to support the development of two exemplar smart living systems.

II. Motivation

Consider the example from Figure 1 of a 'smart home' to assist ageing people. Such a smart living solution aims to provide ageing people with support for physical and mental challenges as they age, but want to stay in their own home longer and be safe and secure [9]. Key features of this system – with a focus on the data-intensive system aspects – include, but are not limited to:

- Sensors that detect the ageing person's movements throughout the smart home – data generated needs to be stored, processed to determine regular behaviour patterns; deviation from these can alert carers that the elderly person is unwell, has a fall, etc.
- Sensors that detect device usage and activity e.g. stove, lights, tap, fridge, etc – as above, this data is integrated, possibly with movement data, to help build the occupant's behaviour profile. It may also be used to support health-related reminders e.g. "Have you had a glass of water
recently?"
- Health-related data capture via e.g. Bluetooth enabled scales, blood pressure monitor, wearable, augmented medicine container, etc – these can check the elderly person is e.g. taking medication, there are no serious outlier health indicators, and data can be shared with remote clinician.
- Communication devices to connect to family, carer and community, such as tablets, smart TVs, etc – these also may provide important behaviour monitoring, but also support connectedness and combat loneliness and anxiety especially of living alone. – Smart home features such as heating, lighting, windows, doors, etc., – these support environment management especially for physically challenged individuals.

Other smart living devices may be added or removed or reconfigured. Other data analysis may be carried out with multiple data input sources. Data privacy, physical and electronic security, and reliability are all critical requirements in this domain.

Smart cities have become a greatly increasing area of research but also of practice [10]-[12]. Consider the example from Figure 1 of a smart city solution which includes diverse data feeds to assist urban precinct operation and strategic planning. A local government instruments a wide variety of its artefacts to assist in providing services and in overall planning and management of services. These might include but are not limited to:

- Smart parking – parking spaces and parking buildings indicate free/occupied spaces. These may be used to provide real-time parking space availability to users, overstaying vehicles to traffic wardens, and over time large scale parking usage statistics for urban planners.
- Smart lighting – which adapts to both ambient light and weather conditions to save power, but also usage information to reduce lighting when unused or increase on demand as pedestrian movement is detected.
- Smart rubbish bins – these proactively monitor rubbish bin space usage, smells, liquids and ask for collection when needed. They also can give long term indications of space usage for planning purposes.
- Space usage sensors – these detect pedestrian and vehicular traffic and can be used to inform different usage patterns according to date, time, correlation to events etc.
- Traffic flow monitoring and control – these provide very detailed information about road usage, and may provide highly localised and adaptive control of vehicle traffic lights and pedestrian crossings.
- Pedestrian activity including movement, density, use of communal tables, chairs, benches, etc – these allow low level monitoring and even proactive intervention e.g. to help enforce social distancing for COVID-19, along with larger-scale data analysis for the usage of footpaths, crossings, communal facilities, blockages etc.
- Building usage and control – these provide the ability to optimise building utility usage to reduce costs and environment impact, and larger-scale data on usage, access needs etc.
- Park management including smart watering systems
- Smart utility management

Local government is continually looking for ways to better manage complex resources and balance demands of retailers, pedestrians, vehicle users, manufacturers, householders etc.

Smart living solutions such as the above provide both low-level, on-the-spot ways to monitor and control government facilities, but also larger scale over time and distance analysis of usage patterns and demand [13]. Unlike the smart home scenario, the users are very large in number and very diverse. The technologies are also more diverse, needing a wider range and scale of platforms and networks to achieve. Privacy and security issues still present but are both individual and group challenges. Scaling to thousands of devices with very large data capture, storage and processing become major challenges.

In both of these systems, a range of stakeholders and developers are needed to develop, deploy and maintain the solution. These range from customers and end users, managers, data scientists, software engineers and edge- and cloud-platform engineers. Sometimes a very wide range of these stakeholders are needed producing a wide variety of heterogeneous devices, servers, networks, dashboards, interfaces, etc. A critical need is for a knowledge management approach that can support this diverse team.

III. Approach

To design data-intensive systems such as those above, we developed the BiDaML suite of visual languages to model key knowledge required [8]. BiDaML provides a set of domain-specific visual languages using five diagram types at different levels of abstraction to support key aspects of big data analytics. These five diagram types cover the whole data analytics software development life cycle from higher-level requirement analysis and problem definition through the low-level deployment of the final product. These five diagrammatic types are:

- *Brainstorming diagram* provides an overview of a data analytics project and all the tasks and sub-tasks that are involved in designing the solution at a very high level. Users can include comments and extra information for the other stakeholders;
- *Process diagrams* specify the key analytics processes/steps including key details related to the participants (individuals and organisations), operations, and conditions in a data analytics project capturing details from a high-level to a lower-level;
- *Technique diagrams* show the step by step procedures and processes for each sub-task in the brainstorming and process diagrams at a low level of abstraction. They show what techniques have been used or are planned to be used and whether they were successful or there were any issues;
- *Data diagrams* document the data and artifacts that are produced in each of the above diagrams at a low level, i.e. the technical AI-based layer. They also define in detail the outputs associated with different tasks, e.g. output information, reports, results, visualisations, and outcomes;
- *Deployment diagrams* depict the run-time configuration, i.e. the system hardware, the software installed on it, and the middleware connecting different machines to each other for development related tasks.

Figure 2 shows an overview of our BiDaML framework in the left side and its notations on the right side. Users including domain experts, business owner, data scientists, and software engineers brainstorm, design and analyse the problem and requirements by defining tasks through visual-based drag-anddrop of the notations through a brainstorming diagram. Business owners and business analysts with the help of the other users would then create a process diagram by specifying the details related to the organisations and participants involved in the project. Users can generate reports and iterate through the different aspects of the projects in order to come up with a plan for the project. Data analysts and data scientists will then focus on data, feature and model engineering parts of the project by further breaking down the tasks to more detailed tasks and connecting them to the data items and techniques. Data analysts and data scientists can generate Python code or connect to the ML tools' APIs in this step. They can embed their Python code to reuse in future projects in this step. Once models are created and finalised, data analysts and data scientists can work with the software engineers to walk them through the models, how they work, the input and outputs of the models and the requirements to access the models.

In the following sections, we use the two representative data-intensive system examples above to show how BiDaML is used to capture and represent key knowledge needed to successfully build each of the systems.

IV. High-Level Requirements Capture

A. Brainstorming

A brainstorming diagram is used to capture the key goals of a data-intensive system expressed at a high-level. There are no rules as to how abstractly or explicitly a context is expanded. The diagram overviews a system in terms of the specific problem it is associated with, and the tasks and subtasks to solve the specific problem. It supports interactive brainstorming and collaboration between interdisciplinary team members to identify key aspects of a system such as its requirements implications, analytical methodologies, and specific tasks. Figure 3 shows the brainstorming diagram for the smart city system.

In Figure 3, the root node, illustrated as a red hexagon, represents the data analytics problem the project aims to address, i.e. the smart city project. High-level tasks are illustrated as orange ellipses with fixed pins to highlight their importance. These include nodes labelled *Define objectives*,

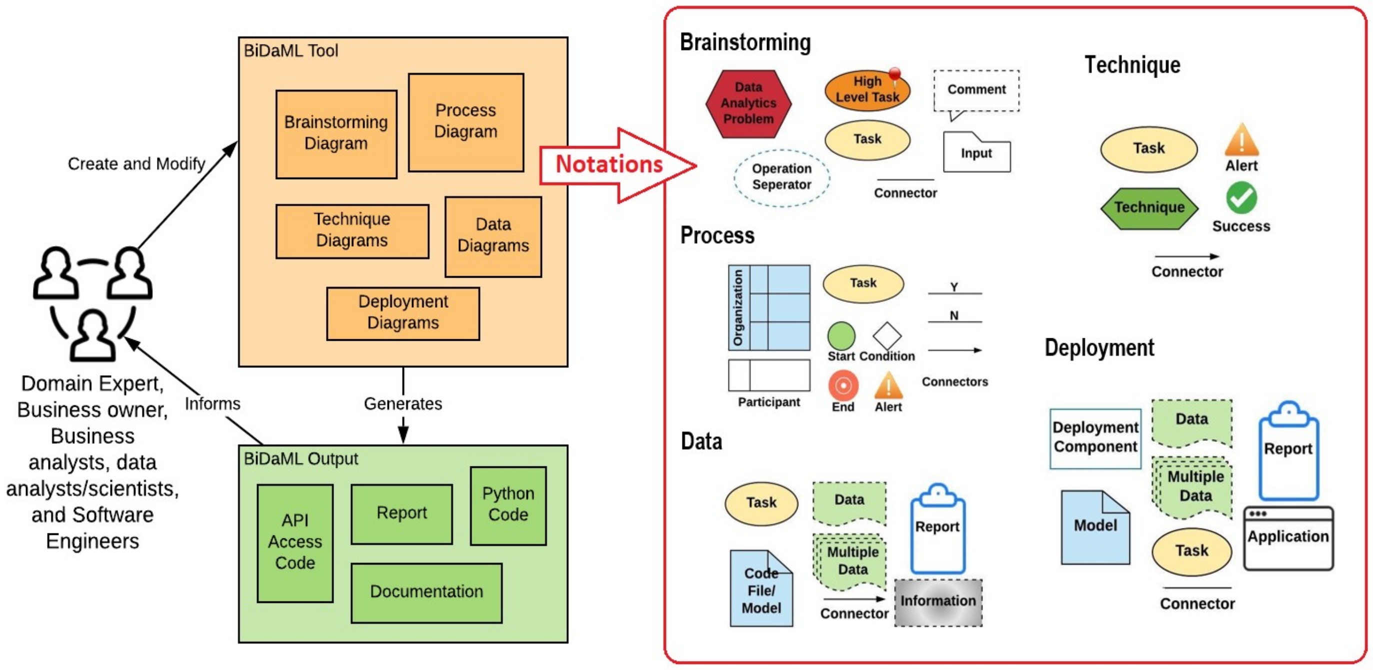


Fig. 2. BiDaML Overview and notations

children of high-level tasks, are called *low-level* tasks. These are drawn using yellow ellipses, such as *KPIs for evaluation* and *Collect data*. A task (both high- or low-level) may be labelled with useful information such as *input* files or *comments* for carrying out the requisite tasks. For instance, the low-level task *Collect data* is labelled by the input file *input.csv*. Comments are illustrated as dialogue boxes with a broken border.

Figure 4 shows the brainstorming diagram for the smart home system. A red hexagon icon represents the Smart Home system and the orange ellipses show the five key to-do tasks with which the problem is associated. High-level tasks connected directly to the problem are automatically changed to bright orange notes with fixed pins to highlight their importance. In this example, *Monitor Activity* is one of the key high-level tasks which is further broken down to *Sensor Data*, *Appliance Data*, and *Health Data* capture and processing tasks as lower-level to-do tasks to achieve this. An input icon representing the *Activity Data* is shown to be produced by this *Monitor Activity* feature. Other key tasks that make up this example smart home include *Health Monitoring*, support for *Anxiety Monitoring*, supporting *Connectedness* to family, friends and carers, and providing *Alerts* e.g. if collected and processed behaviour data indicates the elderly person is unwell.

These brainstorming diagrams provide a high-level conceptual model of the overall system requirements. We drill down from the tasks, subtasks and datasets to define more details of the solution.

B. Process Definition

A high-level process diagram for our smart city use case is shown in Figure 5. In this diagram type, we use an adapted BPMN process diagram representation that captures (i) key organisations as "pools" (blue boxes); (ii) key stakeholders involved as "swimlanes" (white boxes); (iii) key process tasks as yellow ovals; (iv) process start (green circle) and stop (red circle) points ; (v) decision points (diamonds) ; (vi) and process task flows links. The idea is to capture a range of high level, key data processing steps in the system.

For this example, the smart city project starts when the team leaders from the city council share domain knowledge and the existing solutions as well as the parking data with the other stakeholders (top pool/swimlane). The project ends when the engineering team (bottom pool/swimlane), as the technology stakeholders, implement the required sensing technologies. These diagrams clarify the different responsibilities of stakeholders. In this example, the team working with the council and technology stakeholders is made up of several individuals with different responsibilities on the project. This includes *Platform design*, *Platform integration*, etc tasks. A decision point is modelled shown by diamonds with Y/N connectors specifying different conditions that can change the order and priority of the tasks performed. In this example, whether the platform is ready based on the results of the *Testing* task.

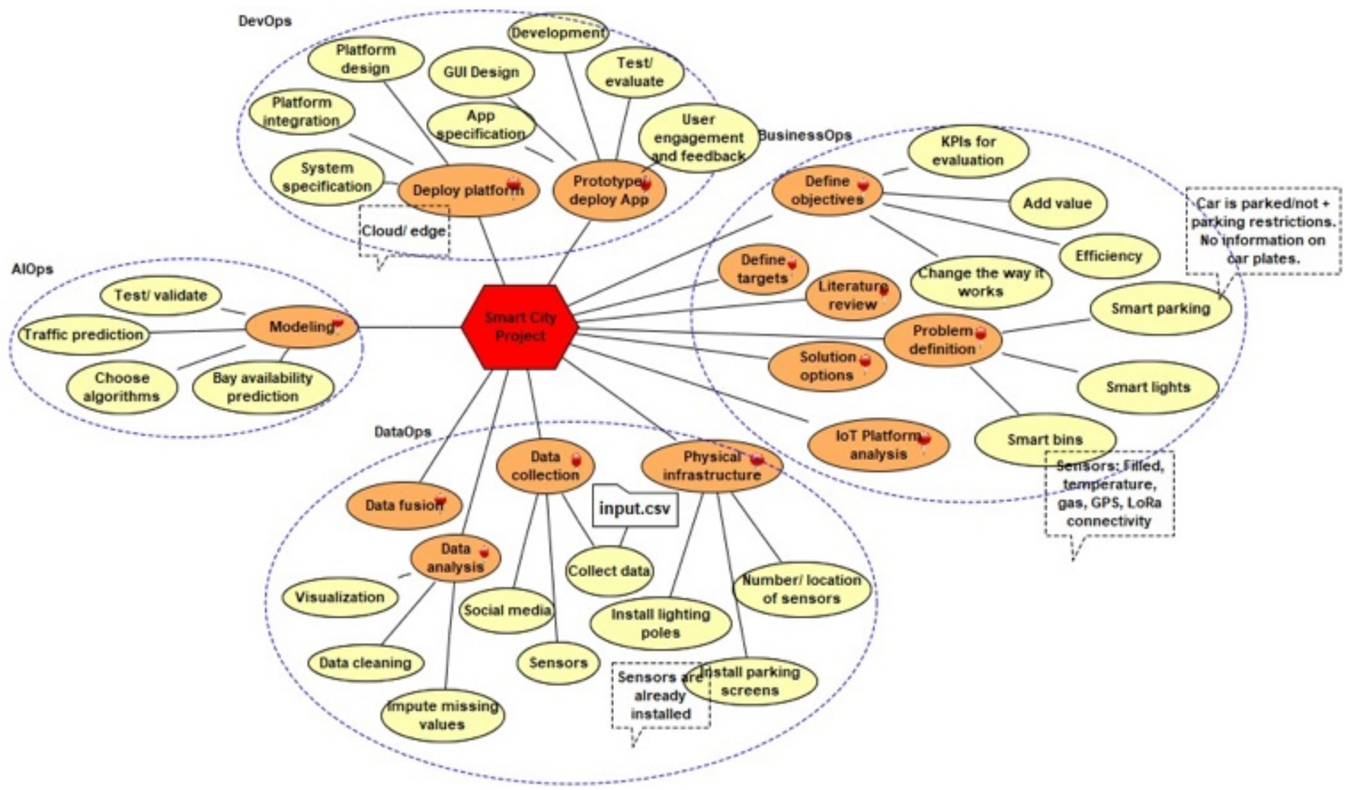


Fig. 3. Smart city brainstorming diagram

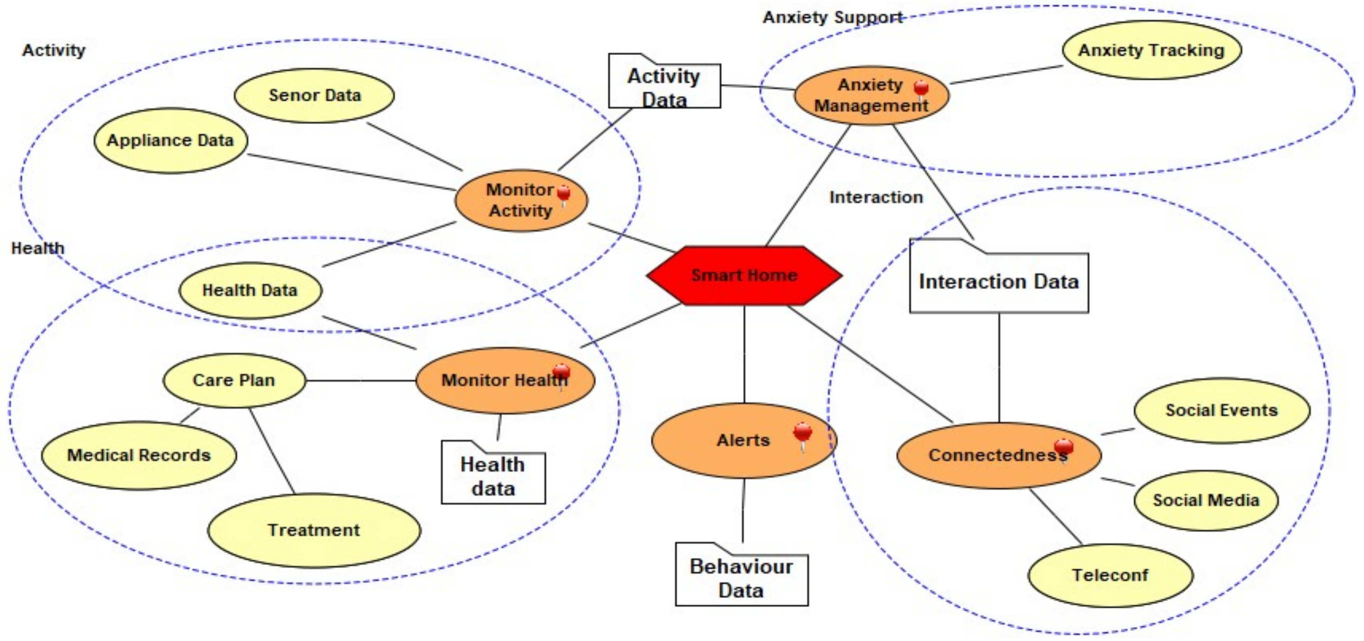


Fig. 4. Smart home brainstorming diagram

For a different usage of these process diagrams, Figure 6 shows a process diagram for part of the smart home scenario. Here, the clinician identifies a need for a smart home configuration for the elderly person. A "care plan" is set up for them which defines the monitoring to be undertaken, the devices to be used, and key tasks used to realise the monitoring. The smart home provider engineers do appropriate installations, configurations and device log monitoring. The elderly user – possibly with assistance – may set some preferences around the devices used, data captured, and give consent. Devices capture appropriate data and store it. Family and friends communicate with the elderly person via

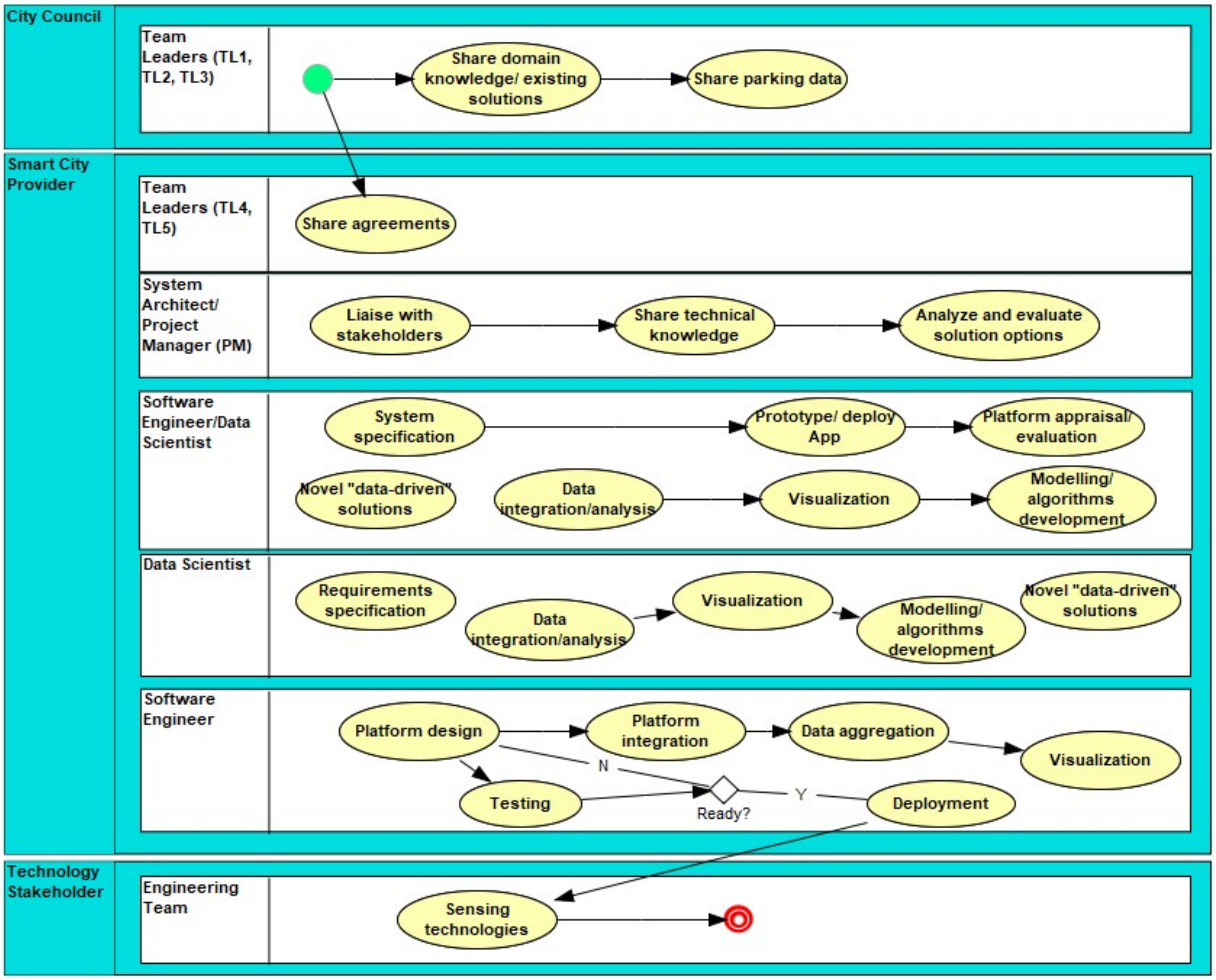

City Council
Team Leaders (TL1, TL2, TL3)
Share domain knowledge/ existing solutions
Share parking data
Smart City Provider
Team Leaders (TL4, TL5)
Share agreements
System Architect/ Project Manager (PM)
Liaise with stakeholders
Share technical knowledge
Analyze and evaluate solution options
Software Engineer/Data Scientist
System specification
Prototype/ deploy App
Platform appraisal/ evaluation
Novel "data-driven" solutions
Data integration/analysis
Visualization
Modelling/ algorithms development
Data Scientist
Requirements specification
Data integration/analysis
Visualization
Modelling/ algorithms development
Novel "data-driven" solutions
Software Engineer
Platform design
Platform integration
Data aggregation
Visualization
Testing
N
Ready?
Y
Deployment
Technology Stakeholder
Engineering Team
Sensing technologies

Fig. 5. Smart city process diagram

appropriate device(s). Some alerts are modelled - the clinician is alerted to health data parameters indicating a possible health issue. Engineers are informed of device malfunction. Friends are alerted when an elderly person misses a requested or scheduled connection. Some of the tasks are ongoing, such as *Monitor Health* and *Monitor Logs* and therefore, only have an 'N' arrow, or are actually intermediate start/end events such as *Initiate Talk Request* and *Accept Talk Request*.

Multiple process diagrams can be defined and can capture different perspectives on data-intensive system requirements. Some process diagrams may be very high-level, while others focus on more detailed processes and tasks within the system.

V. Design

A. Data Management

Data is a critical component of all data-intensive systems. Data is sourced, typically from a wide range of other systems, devices, users, and existing datasets. New data is synthesised when wrangling, integrating and harmonising diverse data sets. New data is also synthesised when applying data processing algorithms within the system. A BiDaML data diagram extends a task notation from the process or brainstorming diagrams with additional attributes or labels about data. This can include sources of data, algorithms used to manage data, and the key producers and consumers of specific data within the data-intensive system.

A sample data diagram for our smart city project is shown in Figure 7. The task icon is reused from the brainstorming and process diagrams to show key producers/consumers of data. Data is represented as green dashed icons, models/code as blue icons, and reports as "clipboard" icons. In this data diagram, we focus on the algorithms associated with the smart parking subsystem. Here the task node titled

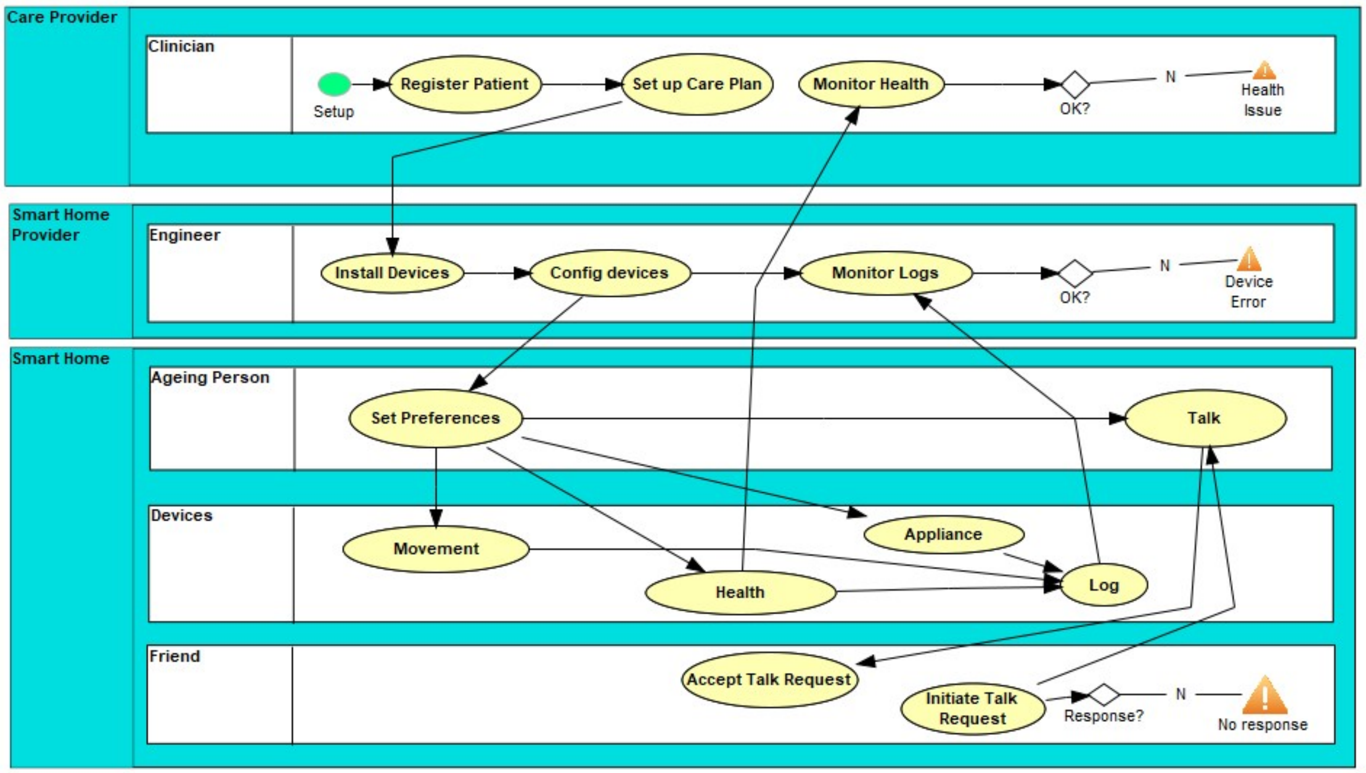
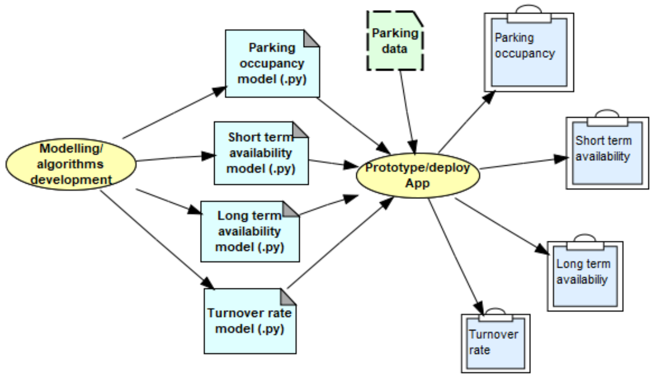


Fig. 6. Smart home process diagram

"Modelling/algorithms development" represents a project task relating to several models that need to be produced to support smart parking facilities. This task generates different models such as *Parking occupancy model*, *Short term availability model*, *Long term availability model*, and *Turnover rate model*. These models are then fed to the *Prototype/deploy App* task, which uses them to generate the corresponding data and reports to show on a smart parking app.



Modelling/ algorithms development
Parking occupancy model (.py)
Short term availability model (.py)
Long term availability model (.py)
Turnover rate model (.py)
Parking data
Prototype/deploy App
Parking occupancy
Short term availability
Long term availabiliy
Turnover rate

Fig. 7. Smart city data diagram

A data diagram used to model the movement monitoring aspect of the smart home is shown in Figure 8. Here we use the data captured from the movement and appliance sensors installed in the smart home to formulate a "behaviour model" over time of the elderly person. This behaviour model is then used by the alerting subsystem to determine anomalous situations that may require informing the elderly people themselves e.g. missed a meal or medication; their friends and family e.g. less movement than expected, missed meal or delayed activity indicating perhaps anxious or unwell; and carer or emergency services e.g. had a fall, no movement at all, in darkness but not gone to bed etc.

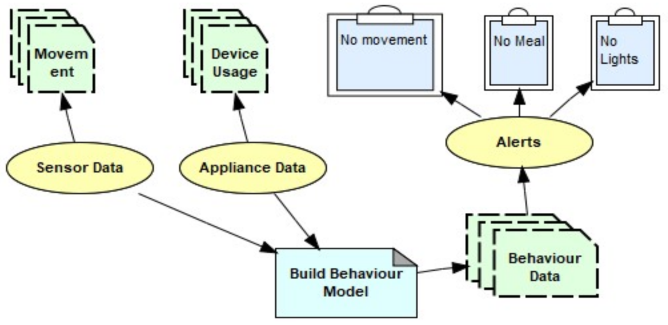


Fig. 8. Smart home monitoring data diagram

With brainstorming and process diagrams, we can define data diagrams at varying levels of detail, define multiple data diagrams for different aspects of the system, and drill down further into processes in terms of the way the data is obtained, wrangled, processed, stored etc.

B. Data Processing

We define a number of technique diagrams to specify the details of how data is processed using BiDaML. Technique diagrams capture the choice of data transformation, wrangling, harmonisation, integration and processing approaches used in a data-intensive system. This includes the choice of feature selection, outlier identification, data preparation for input into a third-party library or package, post-processing extraction of data items, and choice of data analysis algorithms. Yellow ovals again represent tasks from brainstorming, process and/or data diagrams. Green hexagons represent technique choices and how these are chained together to implement the data processing required. Occasionally, techniques used produce errors, which are flagged by suitable alerts.

Figure 9 shows a technique diagram for the smart city system again focusing on the smart parking aspects. In this diagram, *Average parking duration*, *Bay occupancy*, and *Turnover rate* are some example analyses used to realise the *data analysis*. The alert and tick icons attached to the techniques show whether these methods were useful or there were any issues in adopting them. We can create such diagrams for every task and sub-task in the brainstorming and process diagrams as needed. The techniques can be further broken down into sub-techniques used.

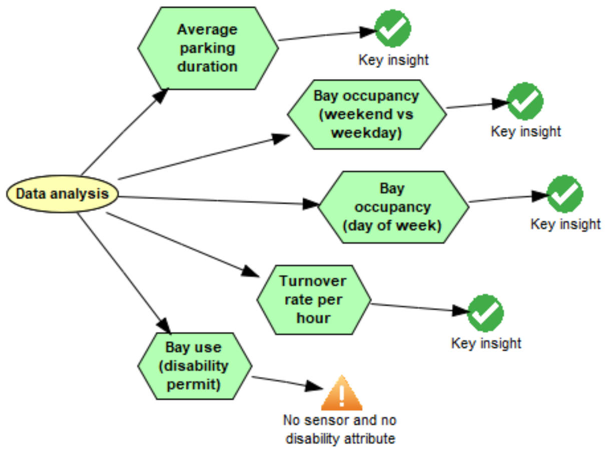

Average parking duration
Key insight
Bay occupancy (weekend vs weekday)
Key insight
Data analysis
Bay occupancy (day of week)
Key insight
Turnover rate per hour
Key insight
Bay use (disability permit)
No sensor and no disability attribute

Fig. 9. Smart city technique diagram

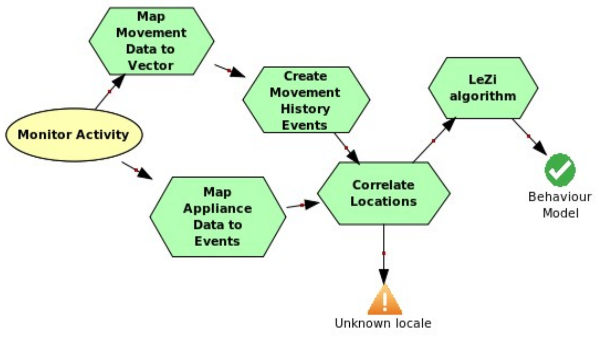

Map Movement Data to Vector
Create Movement History Events
LeZi algorithm
Monitor Activity
Behaviour Model
Map Appliance Data to Events
Correlate Locations
Unknown locale

Fig. 10. Smart home behaviour model technique diagram

Figure 10 shows a technique diagram describing how the behaviour model will be built up from sensor and appliance usage information. In this technique implementation, we build up a movement event history model and correlate this with appliance usage and key locale information from within the smart home e.g. kitchen area, bathroom(s), bedroom(s), living room etc. We then use a Markov model to predict behavioural event sequences, durations and locations. This model is used to check if observed behaviour e.g. movement (or lack of), appliance usage (or lack of), etc do not correspond to the prediction of the model.

VI. DEPLOYMENT

We adapt the UML deployment diagram concept to describe how our data-intensive system components will be deployed in the field. We use BiDaML data, report and task icons to illustrate key locations in the deployment scenario. Devices, servers, networks etc can be modelled. Multiple deployment diagrams can be used to define complex systems from different perspectives.

Figure 11 shows a part of the deployment diagram for the smart city project. Vertical stacking denotes layering of *infrastructure nodes* that describe the technology stack, e.g. a general computing infrastructure is used to run scheduler and workflow monitor that supports the Jupyter notebook server. Horizontal linkages, such as the Restful API/ODBC edge between the Service and Browser or *Mobile*, allow interactions between different technology stacks. Task nodes and attributes from other diagrams are mapped to infrastructure nodes that host them. For instance, the task node "Train models" and the attribute "Parking data" have incoming edges from the service node.

Figure 12 shows an example deployment diagram for part of the smart home system. A smart home "edge server" – represented as a blue box – is used to communicate with the variety of sensors and interactors in the home. This also hosts several applications – represented as rounded rectangles e.g. *Behaviour Modelling*, *Person Reminders*. Data and models are stored on the edge server e.g. the synthesised behaviour model and raw activity, health and interaction data. A personal care plan for the elderly person is kept, developed by their clinician via their *Care Plan Client* application. A provider server hosts various servers and also holds obfuscated data about activities and health, used for population analysis purposes. Data from electronic medical records is obtained from e.g. an EPIC system.

VII. TOOL SUPPORT

BiDaML is equipped with an integrated design environment for creating its five diagrams. BiDaML tool support aims to provide a platform for efficiently producing BiDaML visual models and to facilitate their creation, display, editing, storage, code generation and integration with other tools. Once all the diagrams are created and connected, users can obtain outputs and share them with other stakeholders. There are currently two sets of outputs generated from the diagrams. First, a hierarchy of the graphs can be exported to

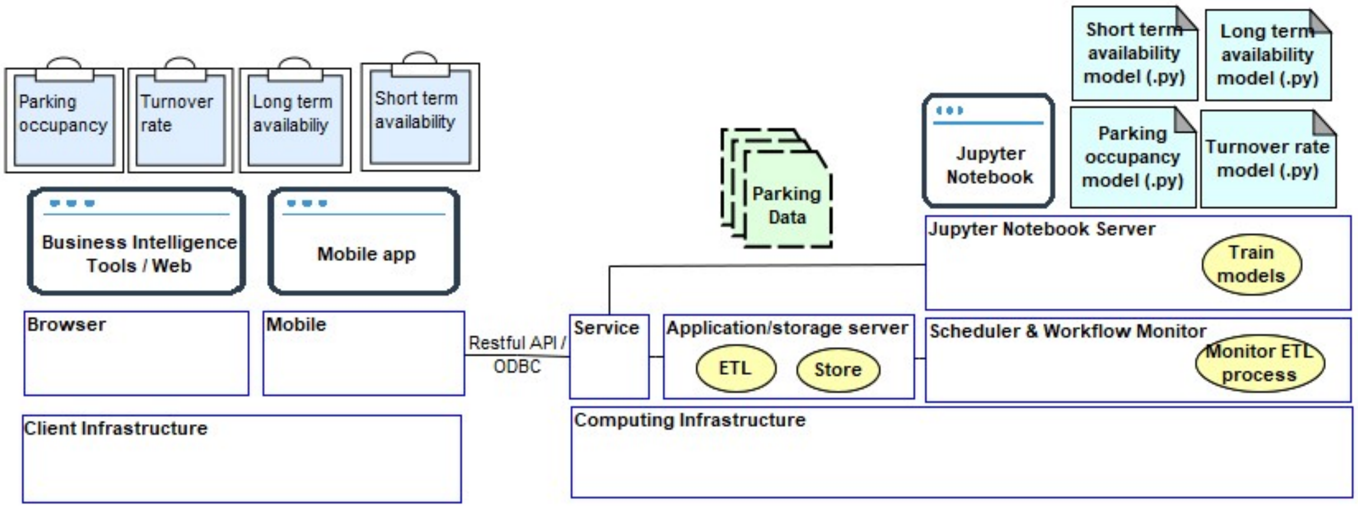


Fig. 11. Smart city deployment diagram

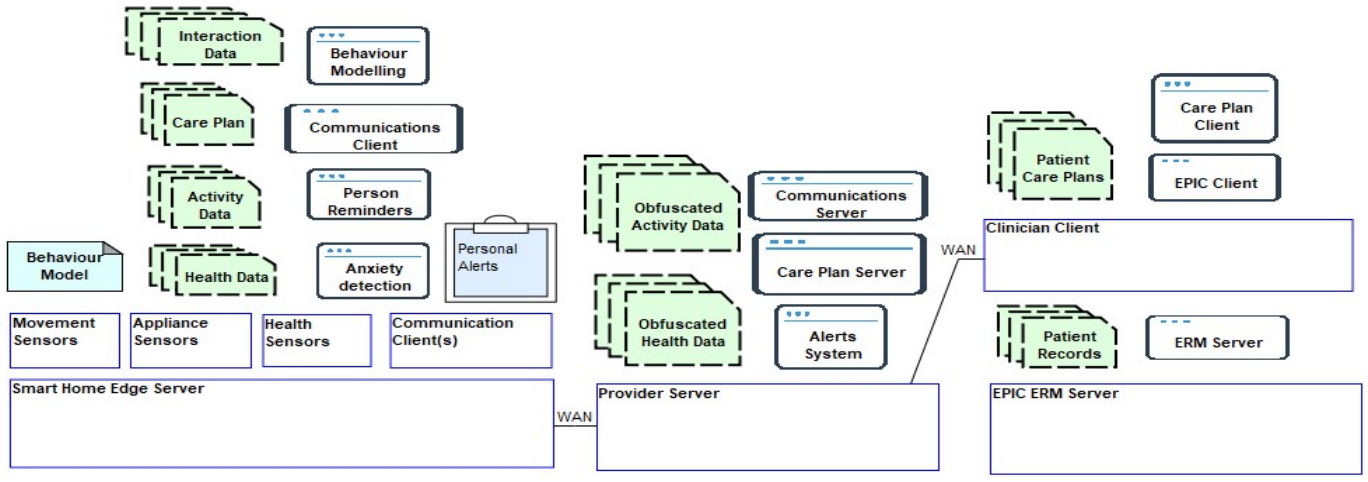


Fig. 12. Smart home deployment diagram

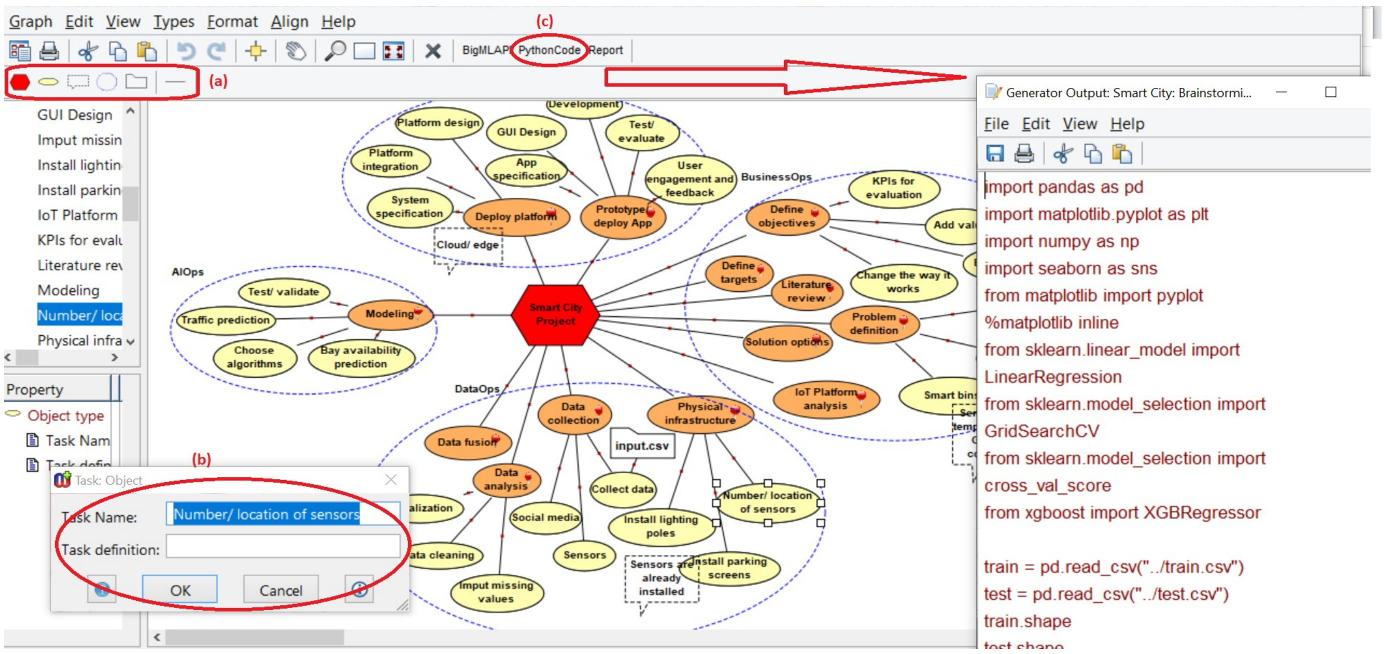
Word and HTML from any of the diagrams. However, since all the subgraphs are connected together in the overview diagram, the most comprehensive report can be generated and exported to Word/HTML through the overview diagram. The second set of outputs are Python code/BigML API, and reports that are embedded in the tool and can be traced back.

An example of the tool used for creating the brainstorming diagram for the smart city project as well as a template code generated for this example is shown in Figure 13. In this figure, users can (a) drag and drop notations, (b) double-click on the notations to rename or modify them, and finally (c) generate template Python code. Another example of the tool used for generating a report from the high-level diagram including all the diagrams created and how they are connected for the smart city project is shown in Figure 14. A report generated for this example is also shown in the right side. In this figure, users can (a) click on the "Generate" option, (b) choose "Export Graph Hierarchy to Word/HTML" to get a comprehensive report of all the diagrams and their explanations in Word/HTML format.

VIII. Discussion

A. Experience to Date

BiDaML has been applied to some other real-world, largescale applications as well, such as a property price prediction website for home buyers, a traffic analysis project, and in different health-related projects. Our aim was to evaluate and gain experience with applying our knowledge management method to conduct requirements analysis and modelling part of complex data analytics applications. We found that BiDaML successfully supports complex data-intensive software systems in industrial settings and it has been practical to a variety of real-world large-scale applications. It helped communication and collaboration between team members from different backgrounds by providing a common platform with mutual language.



Task Name: Number/ location of sensors
Task definition:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from matplotlib import pyplot
%matplotlib inline
from sklearn.linear_model import
LinearRegression
from sklearn.model_selection import
GridSearchCV
from sklearn.model_selection import
cross_val_score
from xgboost import XGBRegressor

train = pd.read_csv("../train.csv")
test = pd.read_csv("../test.csv")
train.shape
```

Fig. 13. BiDaML Tool Support - Template Code Generation

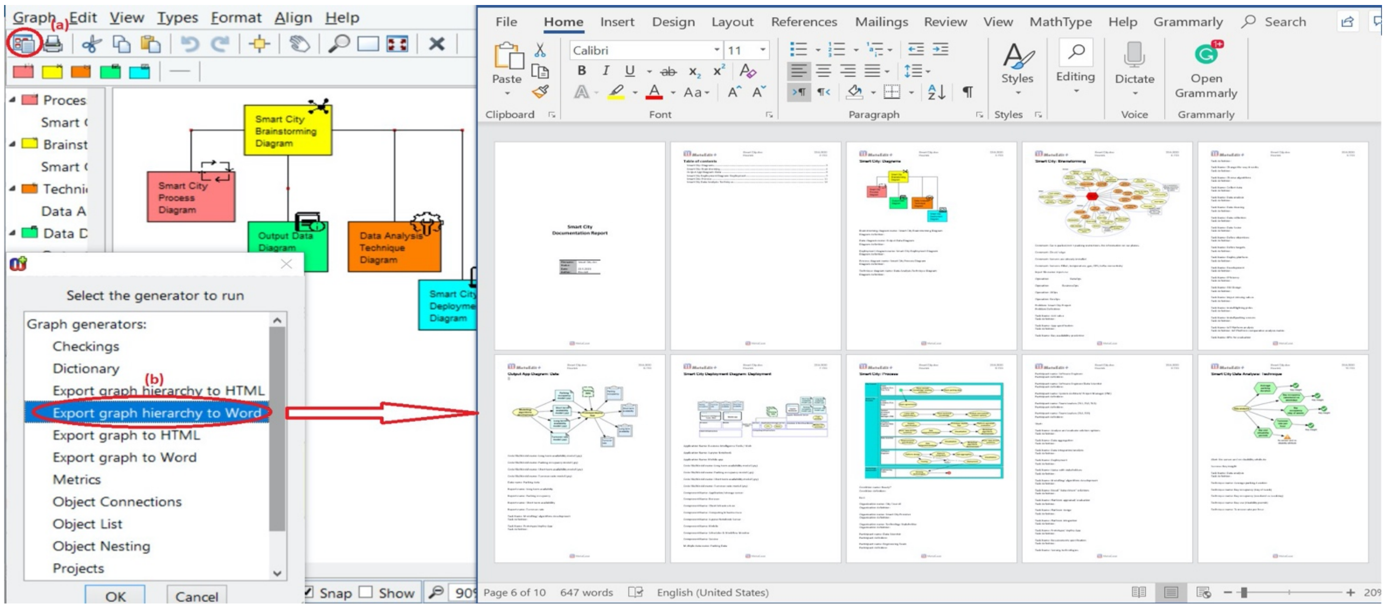

Graph Edit View Types Format Align Help
(a)
(b)
Proces
Smart
Brainst
Smart
Techni
Data A
Data D
Smart City Brainstorming Diagram
Smart City Process Diagram
Output Data Diagram
Data Analysis Technique Diagram
Smart City Deployment Diagram
Select the generator to run
Graph generators:
Checkings
Dictionary
Export graph hierarchy to HTML
Export graph hierarchy to Word
Export graph to HTML
Export graph to Word
Metrics
Object Connections
Object List
Object Nesting
Projects
OK
Cancel
File Home Insert Design Layout References Mailings Review View MathType Help Grammarly Search
Paste
Clipboard
Font
Paragraph
Styles
Editing
Dictate
Voice
Open Grammarly
Grammarly
Smart City Documentation Report
Page 6 of 10 647 words English (United States)

Fig. 14. BiDaML Tool Support - Report Generation

Moreover, it helped identify and make agreements on details in the early stages and therefore, could potentially help reduce the cost and improve the speed of business understanding and requirement analysis stages. BiDaML Provides automatic documentation that can be re-used for retraining and updating of the models. Based on one of our users' experience: "As the frequency of multidisciplinary, collaborative projects is increasing, there is a clear benefit with the use of (The tool) as a tool for designing data analytics processes. Furthermore, the automatic code generation capabilities of (The tool) would greatly aid those who do not have experience in large-scale data analysis. We do see use of (The tool) in this specific project and would be interested in seeing its results."

Based on our two smart home and smart city examples, users have the freedom to design BiDaML diagrams in different ways. For example, the smart city process diagram describes the design of the whole solution, whereas the smart home process diagram describes just the participant onboarding/monitoring process (e.g. there is no Software Engineer or Data Scientist involved). Process diagrams were initially designed in a way to cover all the steps and processes within all the organisations and participants, however, our experiences with these use cases showed that process diagram can be used in either manner (i.e. for development of the whole system, or just for documenting a process). Moreover, in developing the smart home brainstorming diagram, we had the issue of tasks that are ongoing (Monitor Health, Monitor Logs) so only have an 'N' arrow, or are actually intermediate start/end events (Initiate Talk Request, Accept Talk Request). Therefore, in order to reduce the number of symbols needed in BiDaML, informal usage of the intermediate events allows different users to use and specify diagrams based on the preferences (i.e. unlike BPMN we just have start/end/alert rather than a full notation for formally specifying all the different kinds of intermediate events).

There are some notable challenges we faced while working with industrial partners on these data analytics requirement engineering problems. 1) Our tool can be accessed by all the stakeholders in different geographical locations. However, our intervention has been required so far, as the current tool depends on MetaEdit+ modelling development tool [14] and a licence required to be purchased by users; 2) Users make benefits of the early requirement engineering part, however, they continue using existing tools or programming language to develop the ML and application development parts once they have completed the requirement analysis, modelling and planning part of the project. To overcome the first challenge, we are currently working on re-implementing the tool as a stand-alone web-based tool that users can work on individually without any help from us. To overcome the second challenge, we aim to develop integration to popular existing tools to encourage users to continue using our approach through the entire development of the final product. We see considerable scope for providing back end integration with data analytics tools such as Azure ML Studio[1], RapidMiner[2], KNIME[3], etc. Our tool can be used at an abstract level during requirements analysis and design, and then connect to different tools at a low-level. Therefore, BiDaML DSVLs can be used to design, implement and control a data analytics solution [15].

B. Evaluation

We have evaluated our BiDaML approach and toolset from three perspectives:

- As above, we have applied the approach and toolset to model several real-world industrial projects.
- A symbol-by-symbol evaluation of the cognitive effectiveness of the visual notation against established theoretical principles. This was performed using the Physics of Notations framework [16].
- Two user research studies with data analysts/scientists and software engineers under controlled conditions.

Physics of Notations: The BiDaML notations reuse and adapt concepts and notations from Statistics Design Language [17], BPMN [18], and UML [19]. However, we adapted these existing notations to ensure a consistent set of notations for BiDaML that were suited to the needs of multi-disciplinary teams of end users across multiple abstraction levels of modelling. In particular, to facilitate cognitive integration of the different diagrams, we reuse some of the same notational elements across the different types of BiDaML diagrams when they share a common concept.

Using the Physics of Notations framework [16] we analysed each symbol in BiDaML to ensure it was visually distinct from other symbols (e.g. different shape and colour), semantically transparent (not likely to be misinterpreted), and represented a distinct concept. After our first evaluation [15], we performed a major revision of the notations and some of the concepts used [8]. Major updates included a better definition of concept meta-models, new improved notational elements for clarity and consistency, simplification of some diagram models and notations, and a variety of tool improvements.

User research studies: Our first evaluation [15] performed a cognitive walkthrough with three data scientists and two software engineers to perform three predefined modelling tasks. Our second evaluation [8], performed using the revised version of BiDaML presented in this chapter, asked 12 target end-users to model a problem from their own domain. In this second evaluation we asked users to create both a BiDaML diagram, as well as a diagram using the notation (whether formal, textual, or ad-hoc) of their choice. This allowed us to contrast the strengths and limitations of BiDaML against other techniques.

C. Strengths and Limitations

Key strengths of our BiDaML approach as evidenced by our evaluations – trials on industry problems, controlled experiments and Physics of Notations-based analysis - include: suitability for a wide range of diverse data-intensive system stakeholders and developers; facilitating communications and collaborations between multidisciplinary team members; usefulness as a high-level brainstorming and data-intensive system requirements capture approach; assisting reuse of data analytics solutions on new projects and problems; and detailed knowledge capture from multiple perspectives of complex data-intensive system domains.

Key limitations include: for some users, concepts and notational representations take some time to understand and use and terminology is different from their domain of

[1] https://studio.azureml.net/

[2] https://rapidminer.com/

[3] https://www.knime.com/

expertise; limitations with code recommendation and code generation in the current toolset; limitations with supporting distributed collaborative teamwork in the current toolset; bridging the gap and supporting traceability between abstraction model specifications and detailed code solutions; keeping implementations consistent with BiDaML models; and recommending the most suitable data analytics techniques or similar solution implementations from BiDaML specification models.

IX. Summary

We have described two contemporary case studies requiring data-intensive systems - a smart home to support ageing people and a smart city solution. Both have many challenges relating to data-intensive system knowledge management. We have described the use of our BiDaML suite of domain-specific visual languages to provide development teams a variety of modelling techniques to address these issues. We have discussed experience to date with BiDaML, its strengths and limitations, and identified a range of key future work directions to address these limitations.

Acknowledgment

Grundy and Khalajzadeh are supported by an Australian Laureate Fellowship FL190100035, and parts of this work have been supported by ARC Discovery Project DP170101932 and ARC ITRH IH170100013.

References

[1] J. Al-Jaroodi and N. Mohamed, “Service-oriented architecture for big data analytics in smart cities,” in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, 2018, pp. 633–640.

[2] A. De Iasio, A. Futno, L. Goglia, and E. Zimeo, “A microservices platform for monitoring and analysis of iot traffic data in smart cities,” in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5223–5232.

[3] J. Grundy, K. Mouzakis, R. Vasa, A. Cain, M. Curumsing, M. Abdelrazek, and N. Fernando, “Supporting diverse challenges of ageing with digital enhanced living solutions,” in *Global Telehealth Conference 2017*. IOS Press, 2018, pp. 75–90.

[4] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, “Mobile edge computing potential in making cities smarter,” *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[5] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, “Intelligent edge computing for iot-based energy management in smart cities,” *IEEE Network*, vol. 33, no. 2, pp. 111–117, 2019.

[6] A. Cleve, T. Mens, and J.-L. Hainaut, “Data-intensive system evolution,” *Computer*, no. 8, pp. 110–112, 2010.

[7] C. A. Mattmann, D. J. Crichton, S. Hughes, S. C. Kelly, and M. Paul, “Software architecture for large-scale, distributed, data-intensive systems,” in *Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004)*. IEEE, 2004, pp. 255–264.

[8] H. Khalajzadeh, A. Simmons, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, “An end-to-end model-based approach to support big data analytics development,” *Journal of Computer Languages*, p. 100964, 2020.

[9] M. K. Curumsing, N. Fernando, M. Abdelrazek, R. Vasa, K. Mouzakis, and J. Grundy, “Understanding the impact of emotions on software: A case study in requirements gathering and evaluation,” *Journal of Systems and Software*, vol. 147, pp. 215–229, 2019.

[10] A. Fensel, D. K. Tomic, and A. Koller, “Contributing to appliances? energy efficiency with internet of things, smart data and user engagement,” *Future Generation Computer Systems*, vol. 76, pp. 329–338, 2017.

[11] L. Mora, R. Bolici, and M. Deakin, “The first two decades of smartcity research: A bibliometric analysis,” *Journal of Urban Technology*, vol. 24, no. 1, pp. 3–27, 2017.

[12] Z. Rashid, J. Melia-Segu` ´ı, R. Pous, and E. Peig, “Using augmented reality and internet of things to improve accessibility of people with motor disabilities in the context of smart cities,” *Future Generation Computer Systems*, vol. 76, pp. 248–261, 2017.

[13] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, “Understanding smart cities: An integrative framework,” in *2012 45th Hawaii international conference on system sciences*. IEEE, 2012, pp. 2289–2297.

[14] J.-P. Tolvanen, R. Pohjonen, and S. Kelly, “Advanced tooling for domainspecific modeling: Metaedit+,” in *Sprinkle, J., Gray, J., Rossi, M., Tolvanen, JP (eds.) The 7th OOPSLA Workshop on Domain-Specific Modeling, Finland*, 2007.

[15] H. Khalajzadeh, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, “Bidaml: A suite of visual languages for supporting end-user data analytics,” in *2019 IEEE International Congress on Big Data (BigDataCongress)*. IEEE, 2019, pp. 93–97.

[16] D. Moody, “The “physics” of notations: toward a scientific basis for constructing visual notations in software engineering,” *IEEE Transactions on software engineering*, vol. 35, no. 6, pp. 756–779, 2009.

[17] C. H. Kim, J. Grundy, and J. Hosking, “A suite of visual languages for model-driven development of statistical surveys and services,” *Journal of Visual Languages & Computing*, vol. 26, pp. 99–125, 2015.

[18] M. Chinosi and A. Trombetta, “Bpmn: An introduction to the standard,” *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 124–134, 2012.

[19] M. Fowler and U. Distilled, “A brief guide to the standard object modeling language,” 2003.