December 2021

A Simulation-as-a-Service (SIMaaS) Cloud Computing Platform for Traffic Management

Zihao Jin
University of Wisconsin-Milwaukee

A SIMULATION-AS-A-SERVICE (SIMAAS) CLOUD COMPUTING PLATFORM FOR TRAFFIC MANAGEMENT

by

Zihao Jin

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Engineering

at

The University of Wisconsin-Milwaukee

December 2021

ABSTRACT

A SIMULATION-AS-A-SERVICE (SIMAAS) CLOUD COMPUTING PLATFORM FOR TRAFFIC MANAGEMENT

by

Zihao Jin

The University of Wisconsin-Milwaukee, 2021
Under the Supervision of Professor Andrew J. Graettinger

Traffic simulation is defined as a tool to replicate the real-world condition, produce the possible scenario, and facilitate scientific decision making, which can be divided as two categories: planning and operational management. However, the unavailability of vendor-supplied software, difficulty of finding sufficient staff, a lack of community awareness are the major factors impede the adoption of advanced techniques in simulation. This dissertation developed a Simulation-as-a-Service platform (SimaaS), based on cloud computing technology, which can seamlessly integrate transportation simulation engines with flexible cloud-based functional modules like network editor, modeling compiler, cloud middleware, and metrics dashboard to reduce the technical burden of staff in traffic management. A mixed integer linear programming-based task scheduler is embedded in the cloud middleware to improve availability of simulation engines. To facilitate the social involvement, the platform breaks down the service life cycle into three stages for the activity modeling of participants and proposed two business models to underpin activities of various stakeholders. Two sample service cases demonstrate the capability and extensibility of the proof-of-concept. The contributions of this study include:

- Architect a SimaaS framework inherent with a business potential to bridge the gap between transportation simulation providers and end-users (i.e., transportation agencies

and consulting firms) at different project stages;

- Develop a fully scalable and flexible cloud-based network editor and compiler to transform various customer needs into network configuration compatible with simulation engines of various types;
- Develop a back-end mechanisms and algorithms for real-time multiple simulation engine deployment and scheduling;
- Summarize scalable technical architects for interactive dashboards for simulation results aggregation, interpretation, visualization, and decision support;
- Identify, design, and implement three critical stages in service platform modelling for stakeholders. Propose business models for service providers and the platform for developing an evolutionary crowdsourced knowledge-base module to benefit the research and professional community continuously; and
- Implement and test the prototype SimaaS cloud-based platform for real-world applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMS	Analysis, Modeling, and Simulation
API	Application Programming Interface
AEC	Architecture/Engineering/ Construction
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CMD	Command Line
COM	Component Object Model
CRUD	Create, Read, Update, And Delete
DCOM	Distributed Component Objective Model
DOM	Document Object Model
DLL	Dynamic Link Library
eTFOMM	Enhanced Transportation Flow Open Microscopic Model
XML	Extensible Markup Language
FHWA	Federal Highway Administration
GMNS	General Modeling Network Specification
GIS	Geographic Information System
GEOJSON	Geographical JavaScript Object Notation
GUI	Graphic User Interface
HCM	Highway Capacity Manual

HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LOS	Level of Service
MOE	Measures of Effectiveness
M&S	Modeling & Simulation
NSF	National Science Foundation
OLE	Object Linking and Embedding
OSM	Open Street Map
PHF	Peak Hour Factor
PIM	Platform-Independent Interaction Model
PCN	Process-Chain-Network
REST	Representational State Transfer
ROI	Return on Investment
RTE	Run Time Extension
SVG	Scalable Vector Graphics
SimaaS	Simulation as a Service
SOP	Standard Operating Procedures
TRL	Technology Readiness Level
TMP	Transportation Management Plan
UML	Unified Modeling Language
UTDF	Universal Traffic Data Format

UTCS	Urban Traffic Control System
WSGI	Web Server Gateway Interface

ACKNOWLEDGEMENTS

Life is a journey full of challenges. The final delivery of the Ph.D. trip is not a terminal – the dissertation but the trajectory--those connected dots, which cover a series of special events such as storms, flash, rain, and incidents. First and foremost, I would like to express sincere gratitude to the major advisor, Dr. Andrew J. Graettinger, for his decisive take-over at a time of urgency. As an old Chinese saying goes, “Even if someone is your teacher for only a day, you should regard him as your father for the rest of your life.”. Despite the relatively short mentorship, the supportive personality and responsive commitment he left on me would last long.

I would like to express my appreciation to Dr. Yue Liu, who bring me to this beautiful campus, to the world of research. He not only taught me to dive into the deep water to acquire classical hardcore knowledge but encouraged me to float up to the surface and observe the real-world demands. Special thanks also go to Dr. Jie Yu, who is impacted by an ongoing storm. I firmly believe when everything passes, we would see a colorful rainbow, just as we did in many collaborations during the past five years.

I want to express my appreciation to the four individuals who participated on my committee: Zeyun Yu, Ilya V. Avdeev, Rani Elhajjar, and Qian Liao, for their much insight and support. Also, several predecessors helped me with the progress of my work. Among these are Dr. Xuesong Zhou (Associate Professor in Arizona State University), Dr. John Leonard (Professor and Executive Dean in Virginia Commonwealth University), Jeff Sponcia (Planning Manager in Milwaukee County Transit System), John December (UWM alumnus in Osher Lifelong Learning Institute), Douglas Levy (business mentor in Milwaukee I-Corps Program). In addition, I also want to express

my gratefulness to all my colleagues and team members: Wenqing Chen, Yun Yuan, Hui Jin, Dahai Han, Weijie Tan, Xiangyong Luo, Shamsi Mosharraf Trisha, Zhong Liu, and Xinyu Liu.

Most of all, I would like to acknowledge the help and love of my parents and family members, my girlfriend, Chunfang Guan. Without them, just as a vehicle departure without origin and destination, finishing this trip would have been impossible, and I thank them with all my heart.

Chapter 1 Introduction

1.1 Background

With the increasing complexity of systems and processes, Modeling & Simulation (M&S) approaches are widely recognized as effective and valuable solutions that allow decision-makers and system engineers to obtain a proper understanding of the system performance at various levels of abstraction and lifecycle stages (Bocciarelli et al., 2019). In the transportation area, traffic simulation-based models describing all modes of transportation are applied routinely by practitioners and researchers to analyze facilities ranging from individual intersections to extensive regional networks starting from the 1950s (Sperling, 2015).

From the practitioners' perspective, toolsets for transportation simulation have been widely adopted for transportation management plan (TMP) generation, which is required by the Federal Highway Administration (FHWA) since 2004 (FHWA, 2004, p. 200). Nowadays, issues related to the next generation of transportation management systems have become critical. For example, according to the report from FHWA, work zones account for nearly 24 percent of non-recurring congestion, or 482 million vehicle hours of delay (the U.S. DEPARTMENT OF TRANSPORTATION, 2019). Besides traffic delay, it also has adverse impacts on traveler and worker safety. In 2017, work zone fatalities reached a high of 799. More than 37,000 people are injured in work zone related crashes each year. (U.S. DEPARTMENT OF TRANSPORTATION, 2019; U.S. Federal Highway Administration; Texas A&M Transportation Institute., 2018a, 2018b). To cope with the challenges, the generation of TMPs needs to be more efficient and effective.

In fall 2019, 40 interviews were conducted to understand the critical issues in the TMP generation process. During this NSF-sponsored Milwaukee I-Corps program, we learned the service mechanism with respect to a TMP project. In the typical life cycle, state and local transportation agencies in the United States usually describe their management needs in government requirements. Transportation engineers in the third-party consultant company take advantage of signal control software, excel templates, simulation packages, as well as engineering judgment to bid for a contract, then conduct the situation analysis and deliver the final traffic management plan. Finally, this plan will be delivered as a guideline for construction engineers to implement. Concluding from the interview feedback, the biggest challenge for those participants is communication during the collaboration and management process. In this regard, better collaboration and management methods such as interactive visualization, map-based organization, social involvement, etc., have been validated to be beneficial in improving team communication. Besides, one interviewer mentioned there is a web-based application named Remix in the public transit area, which significantly reduces the workforce burden for transit planning work (*Remix*, 2020). However, there lacks a user-friendly equivalent solution to assist simulation work in the transportation management field.

Similar to Remix, many cloud-based platforms have been developed as a web service in various fields. In technical views, Web 2.0 integrates collaboration and cloud computing technology and offers synchronization, online access, platform free access, low cost hardware, low cost processing, and theoretically unlimited low-cost storage (Chen et al., 2015). From the business perspective, cloud computing is an information technology service model where computing services are delivered on-demand to customers over a network in a self-service fashion

that is independent of device or location. This service can be customized, scaled up, or scaled down according to the user's needs (Marston et al., 2011). In the microscopic traffic simulation area, an open-sourced simulator named enhanced Transportation Flow Open Microscopic Model (eTFOMM) was developed on top of 40 years of FHWA traffic simulation algorithms and flow theories for the cloud-based traffic simulation service provider (L. Zhang, 2016). However, most practitioners and researchers do not have the related skills to deploy such a framework on the Cloud. Moreover, lacking an explicit and feasible business pattern is another possible reason for such a methodology not significantly impacting the traffic simulation industry.

1.2 Problem Statement

In the age of computerization, digitization, and intellectualization, the Architecture/Engineering/Construction (AEC) industry is a fragmented and dynamic industry because of the complex nature of its domain knowledge, poor mechanisms for information and knowledge sharing, and lack of formal and interoperable knowledge representation (J. Zhang, 2011). In the transportation industry, many toolsets, software, worksheets templates, guidebooks, and standards have been released for decades. The majority of them can only be applied successfully with detailed guidebooks and a group of experienced engineers. In other words, this type of work style creates high barriers from group to group, company to company, and section to section.

While simulation in transportation at the macroscopic, mesoscopic, and microscopic scale with software such as Cube, Vissum, DTAlite, Transmodeler, etc., has existed for over thirty years it remains challenging to establish a workflow to analyze a simulation from a coarse-grained level

to a fine-grained level. On top of the fragmented workflow, manual effort and coordination from institutions or agencies is another critical problem. First, the higher resolution requires more extensive and structurally complicated data, which may cost overwhelming effort in network modeling. Second, setting up a simulation model consumes inordinate resources, which poses more challenges in result delivery and management. Besides the efforts, issues such as: 1) cost/benefit of implementation are unclear; 2) simulation delivery takes too long; and 3) the underlying modeling approaches are not transparent, are the three main reasons that impede the advance of simulation implementation (X. Zhou & Jiangtao, 2015).

1.3 Proposed Solutions

This research proposes a Simulation as a Service framework (SimaaS) utilizing cloud computing technology and Representational State Transfer (REST) concepts to coordinate data, information, and knowledge flow in a cloud-based platform for transportation simulations. The SimaaS platform is a service system where domain service providers and users are loosely coupled, and the coordination between providers and users is more intuitive because of the involvement of service-oriented workflow concepts.

1.3.1 A service framework to facilitate professional services

Publish/subscribe is a typical communication pattern adopted by information and communication science in the late 1980s to handle large amounts of loosely coupled information producers and information consumers. Following the publish/subscribe model, the Web brings to

bear the capability of collaboration and connectivity, utilizing the browser to support and strengthen this communication pattern (Miller et al. 2001).

Publish/subscribe is an appropriate approach to handling a very fragmented domain knowledge dominated by small and medium-sized businesses. Firstly, it is an effective method to gather fragmented content and grow communities. Secondly, small and medium-sized companies from the service request and provider sides need more exposure to reach an agreement. Thirdly, published content accomplishes the first two steps: socialization and externalization from a knowledge-creating perspective. Lastly, the subscription enables users to perform knowledge combinations based on personal needs. (Shang et al., 2011)

The simulation's dynamic aspect requires that the cloud platform is efficient in handling service requests and responses. A well-designed system needs to dynamically represent various roles that a domain participant could play and understand users' individual needs for information and knowledge. More importantly, a collection application should be able to function in real-time to address the requirement.

1.3.2 Application products to improve service efficiency

The experience of a service is closely related to its operation management. The typical service systems could be grouped into three broad headings: pure service, mixed service, and quasi-manufacturing (Chase, 1981). Ideally, a quasi-manufacturing product is the most efficient way to accomplish the service since it is standardized, automatic, and noncontact. However, simulation service entails heavy customer involvement because of the uniqueness among different real-world applications.

Although there are professional simulation software packages to help technicians apply simulation models, the burden from network modeling, data preparation, and model calibration still impedes the broad application of the simulation service. Many guidebooks and standard operating procedures (SOP) are published for mainstream simulation applications. Some experienced technicians used code scripts from languages such as Visual Basic, Fortran, Python, etc., to automate the operating procedures. However, those procedures or scripts only create value in small communities; there lacks a platform to share individual work on the Internet and gather all the contributions together (Shi et al., 2013). Furthermore, many individual projects should be abstracted, standardized, and encapsulated as a ready-to-use quasi-manufacturing product.

1.3.3 Social involvement to harness collective intelligence

It has been mentioned that the simulation industry needs to prove its rate of return on investment to get the fine resolution simulation contract from the request side. There are two reasons why the rate of return on investment is hard to improve: first, SOP-related scripts are only optimized for individual work under a company-level contract. Therefore, an individual technician can hardly benefit from the sharing process. Second, the consultancy businesses mainly aimed at competition rather than collaboration; thus, the SOP innovations only retains in the small group.

However, the exploited market percentage is far from sufficient; the simulation models and simulation service gap remain enormous. Thus, social involvement is promoted to cultivate a lean service culture empowered by a platform business model to harness collective intelligence and improve industry competitiveness. Ultimately, this collectiveness work will likely broaden the perception of traffic simulation, widen the application field, and explore a larger market share for the entire industry.

1.4 Research Objectives

The primary focus of this dissertation is to develop a Simulation-as-a-Service (SimaaS) platform based on cloud computing technology, which can seamlessly integrate transportation simulation engines with flexible cloud-based functional modules like network editor, modeling compiler, cloud middleware, and a metrics dashboard to improve the efficiency and effectiveness of simulation applications in traffic management. More specifically, the expected contributions of this study include:

- Architect a SimaaS framework inherent with a business potential to bridge the gap between transportation simulation providers and end-users (i.e., transportation agencies and consulting firms) at different project stages.
- Develop a fully scalable and flexible cloud-based network editor and compiler to transform various customer needs into network configurations compatible with simulation engines of various types.
- Develop new mechanisms and algorithms for real-time multiple simulation engine deployment and scheduling;
- Develop an interactive map-based tool for professional simulation results aggregation, interpretation, visualization, and decision support;
- Propose an evolutionary crowdsourced knowledge-base module to take advantage of suggestions, contributions, feedback, and expertise of experienced users to benefit the research and professional community continuously; and
- Implement and test the prototype SimaaS cloud-based platform for real-world applications.

1.5 Thesis Outline

The proposed platform supports the complete life cycle of a simulation service for various transportation studies and consultancy levels. Based on the proposed research objectives, this study has organized the primary research tasks into seven chapters. The core of those tasks and their coherence are illustrated in Figure 1.1.

- **Chapter 1 Introduction** outlines the existing problems in the state-of-the-practice and the motivation for this research in transportation simulation and management concerning real-world, technical, and efficiency issues.
- **Chapter 2 Literature review** presents a comprehensive review of relevant research on various forms of simulation for transportation management, including the technical architecture and solution costs. The review focus on identifying the advantages and limitations of those simulation forms, along with potential enhancements.
- **Chapter 3 Simulation-as-a-Service framework** illustrates the framework of a cloud computing platform based on critical issues that need to be considered in the design of a simulation service. It specifies the service participants, their roles and activities at different stages, the fundamental platform components, and the business requirement, aiming to tackle the operational complexities of a simulation project lifecycle for customizable online service development.
- **Chapter 4 Component-based online simulator** illustrates the details about components and architecture needed to build a cross-platform simulator: (1) cloud-based network editor (2) simulation engine compiler and scheduler (3) error handling mechanism, and (4) cloud

middleware.

- **Chapter 5 Platform modeling** illustrates the structure and behaviour of the platform and how end-users will interact with the system—using the Model-Driven Engineering method to model the entity-relationship, data flow mechanism, user interface, and interaction behaviour under scenarios of various views. Due to the crowdsourcing nature of the platform and the concerns of social involvement, this study also discusses the essential business models of incubating online research in the professional community.
- **Chapter 6 Case study** presents the context analysis based on the lifecycle of procedures regarding the selective simulation process on the platform, which might include (1) simulation service development (2) data collection and preparation (3) Base model modeling and calibration (4) alternative analysis (5) simulation output representation and management.
- **Chapter 7 Summary** summarizes the contributions of this dissertation and provides future research directions.

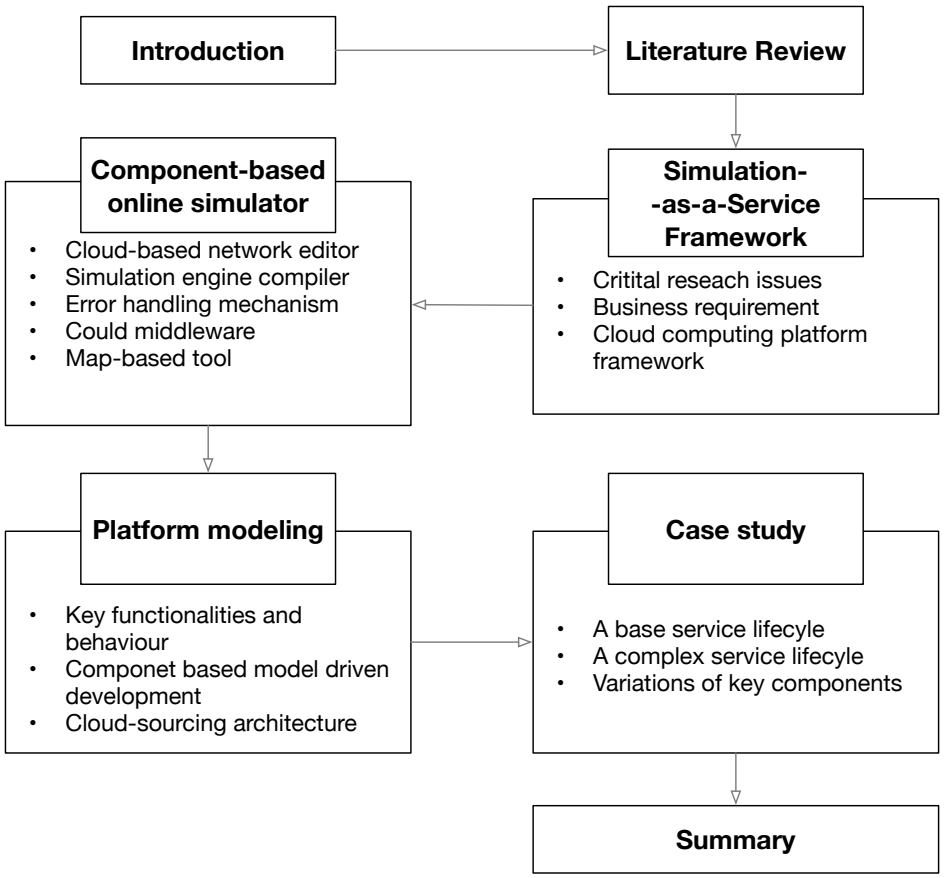


Figure 1.1 The dissertation outline

Chapter 2 Literature Review

2.1 Simulation as a Product

2.1.1 Evolution of traffic simulation applications

The evolution of traffic simulation applications is closely related to the development of digital computers. Backed in the 1950s, simulation model developers had to deal with an adverse computing environment. Possibly the first simulation program in the United States was developed by Harry H. Goode of the University of Michigan (Goode et al., 1956), which simulated the

movement of traffic through a signalized intersection. For demonstration purposes, the model was programmed for the MIADC digital computer. To simulate a single intersection, MIDAC required about 3.2 times as long as real-time traffic flow. Based on the basic concepts of the Goode model, which employs a simulation interval of a quarter-second and treat vehicles on an individual basis, a more advanced simulator, so-called the NBS model, was developed to simulate one particular Washington, D. C. traffic artery containing ten intersections, seven of which were signalized (Stark, 1961). The model was programmed for the IBM 704. To simulate the traffic artery under study, the 704 required roughly 15 times as long as model real time. To simulate a more extensive network with customizable time intervals, rather than simulating in intervals of one-fourth of a second, the Traffic Network Simulator, TRANS, was developed for the District of Columbia to evaluate the effect of traffic signal settings on traffic flow in a region of the city (Katz, 1963). TRANS treats vehicles in clusters rather than individually, and the IBM 7090 took approximately 0.22 times as long as model real time to simulate a network consisting of 268 links.

A new application of simulation technology appeared towards the end of the 1960s in the TRANSYT signal optimization model (Robertson, 1969). Here, in the form of a cycle-based macroscopic simulation model (which took the form of a statistical histogram acted upon by a platoon dispersion formulation), the traffic flow model was embedded as a signal timing iterative procedure component instead of a stand-alone evaluation tool.

While the research community's interest in simulation increased over the decade, most practitioners were either oblivious or dubious of the value of simulation, a posture that extended into the 1980s. E. B. Lieberman and Lou Pignataro demonstrated a potential value of simulation by identifying a problematic signalized interchange. This effort resulted in the DAFT model (E. B.

Lieberman, 1970), coded in the GPSS/360 block language released to the public by IBM, which was used iteratively as a design tool to improve the signal policy when installed at the interchange incrementally. Later on, the FHWA launched a research and development project entitled Urban Traffic Control System (UTCS), which produced a UTCS-1 model to evaluate traffic signal timing policies (FHWA, n.d.). The model used a 1-s time-step with a resolution of 0.1 s, included car following and queuing and lane-change logic, and was capable of simulating much larger networks (E. B. Lieberman, 2014). It is programmed in FORTRAN and is operational on IBM 360/370 and CDC 6600 series computer systems (Carini, 1977). In a small network of 26 links and 12 nodes, UTCS-1 required 5 minutes and 54 seconds of IBM 360/65 CPU time to simulate 15 minutes of real-time (Gibson & Ross, 1977).

Before the birth and wide acceptance of the UTCS-1 model, various simulation models have been developed, tested, or validated at different levels with the emergence of the simulation techniques. The detailed characteristics of popular simulation models are summarized in a table (Gibson & Ross, 1977). The UTCS-1model outperformed its competitors in realism, the flexibility of geometry representation, the extent of model validation as well as simulation speed.

The success of UTCS-1 prompted FHWA to add more features by distributing source code among researchers across the country (Davies, 1972), (E. B. Lieberman & Cohen, 1976), (Carini, 1977), (Gibson & Ross, 1977). The FHWA extended simulation model development to freeways with the INTRAS model (Wicks & Lieberman, 1980). However, the acceptance of simulation was still limited among the practitioner community, primarily because of the considerable procurement cost of large computer systems.

The subsequent PC revolution that happened after the mid-1980s impacted the acceptance of traffic simulation by practitioners significantly. FHWA led the way by porting the NETSIM model to a PC (Sibley, 1985). The animation of simulated vehicle movements was first displayed on PC screens by an FHWA sponsored graphic system, GTRAF (Andrews & Lieberman, 1987). Subsequently, refinement and extension of the INTRAS model for the PC were also sponsored and produced by FHWA, the so-called FRESIM (Halati & Torres, 1990). About the same time, NETSIM has been used by over 130 local and state agencies (Chang & Kanaan, 1990), (Paksarsawan et al., 1992). E. Lieberman (1991) developed a package named HCM/Cinema with an existing software named TransCAD to serve as an interface custom application by combining the strengths of geographical information systems (GIS), microscopic simulation, and computer animation, which is also supported to combine HCM-based analyses with TRAF-NETSIM (McGhee & Arnold, 1997).

In the 1990s, the FHWA revised its simulation tools to satisfy the requirements of ITs applications (Santiago & Kanaan, 1993). Two of its popular microscopic traffic simulators for freeways (FRESIM) and urban networks (NETSIM) have been combined into a more general traffic simulation framework called CORSIM (FHWA, 1996). In the same decade, the transportation industry witnessed an explosion in prominent simulation packages including, but not limited to, Aimsun: Transport Simulation Systems (TSS); CUBE: Citilabs; Dynameq: INRO; MITSIMLab: MIT; PARAMICS: Quadstone; Simtraffic: Trafficware; Transmodeler: Caliper; and VISSIM–VISUM: PTV. Among these products, many provide application programmer interfaces (APIs) so that skilled users can customize these packages using high-level programming languages, developed supporting tools to suit their needs. With the success of commercialization and

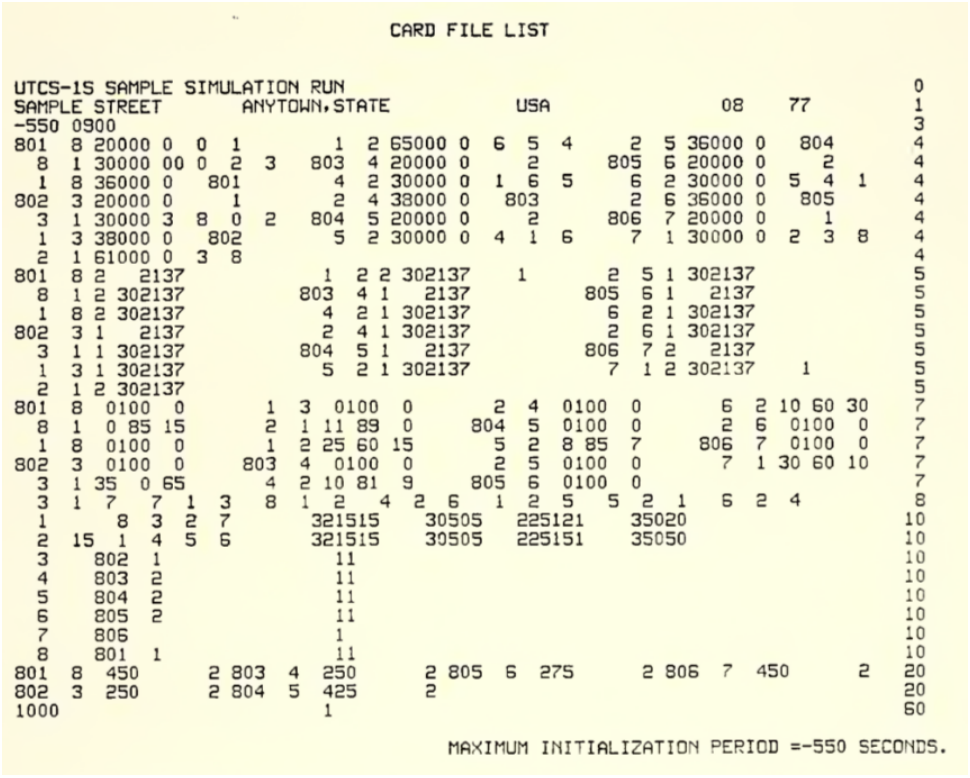
extensibility, all commercial simulation-based products were marketed worldwide, e.g., the PTV VISSIM has about 7000 licenses distributed worldwide in its first 15 years (Fellendorf & Vortisch, 2010).

2.1.2 Simulation network modelling and representation

Typical simulation software comprises three modules: 1) computation, 2) data input, and 3) output representation. First, the computation module is the simulation software engine derived from classical principles such as car-following logic and lane-changing models. Second, the data input module takes the inputs from users and models the inputs such as network geometry, volume in a format that the computation module can read. Last, the output representation module provides the measures of effectiveness (MOEs) of the entire network or a specific link, which could be viewed as graphical or tabular summaries.

The geometric modeling of an existing transportation network is an essential part of preparing for the input and output of the simulator. The TRANS employed a "link," i.e., a directed traffic artery, as its fundamental simulation unit. Links are of three types: input, network, and output. Each link (input type or network type) has a traffic signal associated with it, and the characteristics of each link are given to the program as input (Katz, 1963). The UTCS-1 simulator adopted the link and node representation, which requires a detailed description of the street system, the traffic control option, and volumes as input. As output, the simulator lists link-specific and network-wide MOE's (Carini, 1977). The UTCS-1 input data must follow input requirements according to function type, including Identification cards, Link cards, Signal or Sign control cards, Flow rate cards, Control cards, and Embedded Data Change cards. A sample input card is shown

in Figure 2.1 (Carini, 1977), the CORSIM 6.0 version inherited the 80-column format of the input card.



CARD FILE LIST

```
UTCS-1S SAMPLE SIMULATION RUN                                                    0
SAMPLE STREET       ANYTOWN,STATE          USA              08    77           1
-550 0900                                                                        3
801  8 20000 0  0  1       1  2 65000 0  6  5  4     2  5 36000 0   804         4
  8  1 30000 00 0  2  3   803  4 20000 0     2      805  6 20000 0     2        4
  1  8 36000 0   801         4  2 30000 0  1  6  5     6  2 30000 0  5  4  1    4
802  3 20000 0     1         2  4 38000 0   803        2  6 36000 0   805       4
  3  1 30000 3  8  0  2   804  5 20000 0     2      806  7 20000 0     1        4
  1  3 38000 0   802         5  2 30000 0  4  1  6     7  1 30000 0  2  3  8    4
  2  1 61000 0  3  8                                                            4
801  8 2   2137             1  2 2 302137    1       2  5 1 302137              5
  8  1 2 302137           803  4 1   2137          805  6 1   2137              5
  1  8 2 302137             4  2 1 302137          6  2 1 302137              5
802  3 1   2137             2  4 1 302137          2  6 1 302137              5
  3  1 1 302137           804  5 1   2137          806  7 2   2137              5
  1  3 1 302137             5  2 1 302137          7  1 2 302137    1          5
  2  1 2 302137                                                                5
801  8  0100  0        1  3  0100  0       2  4  0100  0       6  2 10 60 30    7
  8  1  0 85 15        2  1 11 89  0     804  5  0100  0       2  6  0100  0    7
  1  8  0100  0        1  2 25 60 15       5  2  8 85  7     806  7  0100  0    7
802  3  0100  0      803  4  0100  0       2  5  0100  0       7  1 30 60 10    7
  3  1 35  0 65        4  2 10 81  9     805  6  0100  0                       7
  3  1  7   7  1  3   8  1  2   4  2  6   1  2  5   5  2  1   6  2  4          8
  1     8  3  2  7       321515    30505   225121    35020                     10
  2 15  1  4  5  6       321515    30505   225151    35050                     10
  3    802  1              11                                                  10
  4    803  2              11                                                  10
  5    804  2              11                                                  10
  6    805  2              11                                                  10
  7    806                 1                                                   10
  8    801  1              11                                                  10
801  8  450       2 803  4  250      2 805  6  275      2 806  7  450      2    20
802  3  250       2 804  5  425      2                                          20
1000                      1                                                   60

                                   MAXIMUM INITIALIZATION PERIOD =-550 SECONDS.
```

Figure 2.1 List of data check of the UTCS-1 simulator

However, most users were discouraged by the labor-intensive effort of preparing and processing the input data and by the effort needed to shift through extensive statistical output listings to extract the required information. The birth of GTRAF let technician personnel with no knowledge of computers or traffic engineering entered input data effectively through a graphical interaction. The logical structure of the GTRAF Display hierarchy mainly divided network data into two categories according to two steps known as Input data and NETSIM results (Andrews & Lieberman, 1987).

Fast interactive response and ease of a multimedia user interface are the pursuits for urban traffic planners to build a network representation. NEDIT is a data editor for the entry development of the TRAF-NETSIM model. It is arranged as a hierarchy of menus and options that facilitate user entry of input data by leveraging the online help function and the automatic error checking capability (Joan D. Sulzberg & Michael J. Demetsky, 1991). However, if one is familiar with the format of a TRAF-NETSIM input stream, it may be more efficient to edit input data using a word processor rather than NEDIT. Therefore, to better facilitate users entering data, the TRAF-EDIT was developed to enhance the interactivity of the NEDIT input data manager.

On the one hand, it supports the Quick Edit option, which splits the 80-column data into individual data items (entries). This option allows the user to edit data for individual entries while providing all the helpful information. On the other hand, it also has a Smart Edit option that offers online access to the TRAF user guide for the data described on a given line (Rathi et al., 1989).
After the simulation has run, interpretation of the simulation results is most critical. Yang et al. (2006) registered a U.S. patent regarding a system that integrates GIS with traffic simulation processes to allow a user to analyze traffic patterns and loads at specific geographic locations of regions. 3D visualization packages were gradually integrated into systems such as VISSIM, WATSIM, and PARAMIC (Boxill, 2007). The visualization techniques adapted by some widely used simulation tools have been summarized and compared in terms of the method of 3D object generation (B. Zhou et al., 2005). Despite the progress in visualization techniques, especially in the animation field, there still lacks a study on evaluating how such visualization techniques impact decision making. Furthermore, the animation is a form of final presentation that is most beneficial

for technical-to-nontechnical communication. However, a method to improve the connectivity and interactivity of technical-to-technical communication is an area that needs more attention.

2.1.3 Derived simulation applications

The combination of new developments in computing and new problems in practice stimulates new thinking about methods for producing models with which traffic systems can be simulated. From an application design perspective, the more flexible the building process is, the more usable the software would be in real-world applications.

A generalized error message procedure was designed to alleviate the storage demands of diagnostic processing for INTRAS (Wicks & Lieberman, 1980). Chang and Kanaan (1990) applied a batch-means method in TRAF-NETSIM to assess the variability of crucial output parameters of interest under various control strategies. The batch means method, with a given batch size, is relatively inexpensive in computational cost, allowing the estimation of confidence intervals to be completed in a single long simulation run. In terms of the simulator, applications have been developed on the PC platform to facilitate the simulation procedures (pc-trans, 2003). TRAFVU is a graphical animation tool designed to display the system output of the CORSIM simulator. TRAFED is an input processor which allows users to create CORSIM simulation input files much easier and faster. Smart Editor is an enhanced version of TRAF-EDIT to modify the CORSIM TRF file. Besides the applications mentioned above, other features like script tool, run time extension (RTE) (Mears, 1999) were integrated into a Graphic User Interface (GUI) called TShell to manage simulation procedures better and the whole software package named TSIS (Owen et al., 2000). To overcome the limitation of RTE, for example, difficulties in programming, a new type of CORSIM client/server API was proposed to interface CORSIM with a third-party virtual traffic

controller server (L. Zhang, Morales, et al., 2002). On interfacing with the simulation output side, John D. Leonard II developed a Visual Basic script to access CORSIM TSD binary files, storing the vehicle and signal messages by utilizing CORTOOLS.DLL OLE "automation server" in CORSIM (Alvarez Mendoza, 2008). From 2008 to 2017, Zhang Li worked closely with FHWA to develop a distributed/cross-platform application programming interface based on CORSIM through an open-source approach (L. Zhang, 2016).

With the improvement in interface design, APIs between Microsoft applications have evolved from Dynamic Link Library (DLL), Object Linking and Embedding (OLE), to Microsoft Component Object (COM) and Distributed Component Objective Models (DCOM) (L. Zhang, Ghaman, et al., 2002). VISSIM implements COM Model as a programming interface that started in 2003. The functionality provided by a COM interface can be used by different programming languages, among them popular scripting languages like Visual Basic or Python. The COM interface can be used to include VISSIM in other applications as well as simplify and document repetitious simulation tasks. (Fellendorf & Vortisch, 2010). For instance, Lu et al. (2015) developed a simulation platform by integrating VISSIM with C++ and MATLAB through the COM interface to ease the difficulty of secondary development and provide a new way to manipulate events.

The aforementioned derived developments are a few examples of the external application galleries. Most of the application galleries were only used inside a small group; few could be accessed through the open-source code community. The simulation community can barely benefit from practical usage unless the practitioner has a strong background in the theoretical base and

programming skills. Moreover, the cost of software licenses is another barrier for either practitioners or researchers exploring various secondary development methodologies.

2.1.4 Cost-effectiveness of simulation

There are two primary cost types in a simulation service: labor cost and software cost. Dowling (2002) concludes a rough rule of thumb for staffing a microsimulation project. About eight person-hours of effort per freeway interchange (or signalized intersection not at a freeway interchange) of effort per freeway interchange (or signalized intersection not at a freeway interchange) to set up and error checks the coded network. This amount equals about 50% of the budget. Another 25% of the budget is spent on calibration, leaving 25% for alternative analysis, documentation, and presentations. Alexiadis et al. (2014) categorize simulation projects into small, medium, and large models. Small networks, which include 5–20 mi of freeway with three freeway interchanges and signalized intersections on either side of the interchanges, usually takes about 500 person-hours. Medium networks, which include 10–20 mi of freeway with 4–15 freeway interchanges, approximately accounts for 1500-2000 person-hours. Finally, an extensive network, which include 20–40 mi of freeway, 15–35 interchanges, and 1–2 interstate systems, requires over 10,000 person-hours, or five man-years.

Software cost is challenging to evaluate since the different packages have different features and market positions; therefore, they have a difference in the pricing scheme. From a practical perspective, Xu and Han (2012) summarized the features and reference price for popular traffic microsimulation packages. Detailed information is presented in Table 2.1.

		AIMSUN	TSIS CORSIM	PARAMICS	VISSIM	CUBE DYNASIMS	MITSIM	INTEGRATION
Country		Spain	U.S.	U.K.	German	U.S.	U.S.	U.S.
Version		6	6.2	6.4	5.3	-	-	2.3
Modeling Scope		Freeway and urban networks	Freeway and urban networks	Freeway	Urban networks	Freeway and urban networks	Freeway and urban networks	Freeway and urban networks
Language		English	English	English	Chinese, English, French, German	English	English	English
Reference Price		$ 25,000	$ 500	$ 25,000	$ 14,000 (basic network)	$ 25,000	-	-
Network Size	**Link**	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited	-	10,000
	Node	Unlimited	9000	Unlimited	9999 (signal controller)	Unlimited	-	-

Table 2.1 Comparison of Traffic Microsimulation Software Package Costs

2.2 Simulation as a Service

2.2.1 Types of service

Before knowledge is standardized to a product, it was a service in amorphous. Kotler (1980) stratified services into people-based and equipment-based services. Examples of equipment-based services are airlines and vending machines, while people-based services include insurance, banking, and consultancy services (Silvestro et al., 1992). Chase (1981) proposed a customer contact approach and classified services based on the potential facility efficiency. According to decreasing contact criterion, standard service systems could be stratified into three categories: pure service, mixed services, and quasi-manufacturing. Vandermerwe and Rada (1988) further developed the goods-to-services continuum in the following three stages: First, the company is in

either a goods or service business; Second, goods and services are combined in the offerings; Third, offerings are complex bundles of goods, services, information, support, and self-service elements. Oliva & Kallenberg（2003) consolidated product-related services and expanded the installed base (IB) service offering into four quadrants: transaction-based services, relationship-based services, product-oriented services, and end-users process-oriented services. In this context, transportation simulation falls in the range of transaction-based and end-user process-oriented services specifically pointed to professional services. Moreover, the organizational arrangements and offerings align with factors of a service transaction firm (Fundin et al., 2012).

2.2.2 Service practice

No matter how complicated a professional microsimulation is, a simulation is strongly based on engineering judgment that involves stacks of tacit knowledge and face-to-face contact. According to the criterion posed by Oliva & Kallenberg (2003), the traffic simulation service is a process-oriented professional engineering service which is also under the transaction-based services category. A simulation study consists of the following tasks: 1) Formulation of the aims and scope of the study, 2) input data collection, 3) construction of the simulation model, 4) model verification, 5) model calibration, 6) model validation, 7) alternatives analysis, and 8) documentation. In a typical goods-service spectrum, the simulation software business is positioned as core goods with accompanying service; the conventional consulting firm operates in core services mode with accompanying goods. However, the industry fragmentation enhances the professional barriers in such a way that the software becomes more and more complicated, expensive, and hard to interpret, which causes the steep learning curve for a qualified

transportation engineer. Those barriers are not directly related to the current priorities of simulation service but impede the scalability of traffic simulation.

From the technical perspective, the study of traffic conditions using simulation needs multiple instances to be executed for a set of different parameters (Dowling, 2002; Nevada DOT, 2012; Olstam & Tapani, 2011). Shekhar et al. (2016) developed a cloud middleware for SimaaS on a Linux container to generate different configurations for experimentation, and schedules the simulations intelligently. To address the provision of coarse-grained simulation services, which offer the user easy access but rigid orchestration of M&S components consisting of entire environments, applications, and tools. A microservice-based approach for fine-grained simulation services has been developed by leveraging cloud computing technology (Bocciarelli et al., 2019). However, the industry's complex and diverse nature demands effective and human-friendly communication systems because enhancing communication between project stakeholders is the first step in assuring efficient decision-making in collaborative environments (J. Zhang, 2011). Because of the dynamic environment, fragmented, and short-term nature of the individual project, the simulation service has been characterized by low utilization of an advanced communication system. Therefore, a service platform that can interface with a deployed service host in the technical end and advance service progress in the users' view is needed to facilitate collaboration, communication, and innovation between stakeholders. Besides, the service platform should be hosted online and have the crowdsourcing feature to enhance social involvement in the simulation domain.

2.2.3 Service design and implementation

Selling the simulation know-how product online is a type of E-business. Since the Web brings to bear the idea of collaboration and connectivity (Miller et al., 2001), E-business enables transactions to occur online and enhances the operation speed and efficiency of the business transactions, thus reducing costs, time and improving operations. However, implicit knowledge is difficult to convert to a specific product. Implicit knowledge has existed in people-based services for a long time and is used to complete the business process by introducing face-to-face interaction. Unlike e-business, selling knowledge online needs to move to the transactional side of business online. For instance, It can offer client analytics and additional value wisdom for their business processes and operations (Mazen, 2013). Thus, the design of the transactional side of a simulation service affects the effectiveness of its implementation.

From the standpoint of software design, the modular design principle is nearly ubiquitous in developing open-source traffic simulation packages (Behrisch et al., 2011; Byrne et al., 2010; Tamminga et al., 2012). Shifting the design pattern from the developer to the end-user, the MVC (Model-View-Controller), an event-driven design pattern, is adopted to implement a conceptual design for a Research-Oriented Web-based Traffic Simulation Platform (Shi et al., 2013). Even though the adaption of a web framework can improve the portability and interoperability of the simulation model, it remains problematic in simplifying the simulation modeling process. Thus, traditional simulation tools should be developed that interact with web tools and are able to reuse existing simulation models (Wiedemann, 2001). Over and above that, the role of a web front-end is to guide the user by a customized workflow using the principle of business process management. The possible service workflows for the traffic simulation study include, but are not limited to, the

representation of traffic facilities operating close to capacity, representation of networks operating close to capacity, level-of-service analysis (Olstam & Tapani, 2011), real-time simulation systems with a bundle of software tools, knowledge and sometimes know-how, GIS applications to fuse data together from different contexts as a common platform, rural area simulation with heterogeneous traffic and poor data quality, and a simple or automated calibration tool (Pell et al., 2017).

2.2.4 From laboratory to service

The transformation of professional knowledge to an application and service usually involves developing, testing and debugging, deploying, and lean launching. Regarding the development phase, Buschiazzo et al., 2010 developed an interactive remote laboratory that allows developing and executing real C applications in a completely web-based environment. It demonstrates the flexibility and interoperability in the e-learning scenario through a Digital Signal Processing case using a service-oriented architecture. Google integrates the user interface of JupyterLab with a serverless cloud environment to bring up a free product named Colab (Bisong, 2019). It is widely used for machine learning models and prototyping, and collaborative development. In order to further deploy the code in development to a ready-to-used tool service, Transportation Research and Education (ITRE) established a solution-oriented laboratory, Driver and Transportation Analytics Laboratory--DaTA Lab (Vaughan et al., 2017). The Traffic Operation and Safety (TOPS) Laboratory developed many automated services to facilitate data management (Parker & Tao, 2006), whereas the DaTA Lab built online tools applicable in actual practice on top of the data portal. Although both laboratories gathered a collection of cloud tools for selected users, they leave the development process at the ad hoc local desktops. Therefore,

there is a gap between those solution-based laboratories and collaborative development laboratories.

The Federal Highway Administration (FHWA) published a Technology Readiness Level (TRL) Assessment tool for determining the maturity of a technology(Towery et al., 2017). The TRL scale covers four categories: basic research, applied research, development, and implementation nested with steps ranging from Level 1 to Level 9. Until now, most cloud services such as applications, storage, servers, processing power, and databases are in the stages from Level 1 to Level 5, before the Applied Research Phase. However, inexperienced users or customers still encounter challenges in learning, configuring, and using the developed tools on those platforms, which is a major impedance for transportation simulation adaptation to a cloud solution and the advancement from Level 5 to Level 9, towards the Development Implementation phase.

2.3 Data Interoperability

Data collection is one of the critical issues determining the quality of a simulation service since the level of data acquisition effort is highly correlated to the simulation's resolution and scale. It is not a cost-effective solution for the small and medium project to conduct a high resolution analysis and simulation from the practice perspective. Mainstream professional software resorts to internal functions to convert its data format to suit the best compatibility from other collaborators, thus reducing the labor cost of recoding network information. Besides, few data schemes have been established in order to reach a higher level of data interoperability.

2.3.1 Common formats in practice

Many open-source and proprietary file formats exist, which leads to siloed data if the formats are not compatible. DTALite is an open-source lightweight engine designed for mesoscopic simulation. Its essential input files and output files are listed as CSV and TXT files (Hamilton). A user can open the NeXTA, a GIS-like user interface to perform the operations that interact with the CSV files and call the simulation engine's functionalities. It integrates some data conversion tools such as: 1) macroscopic network (e.g., VISSUM, TransCAD, CUBE) to mesoscopic network, 2) mesoscopic network to Microscopic network (Synchro UTD format and VISSIM ANM file).

Highway Capacity Software (HCS) implements the Highway Capacity Manual (HCM) methodologies for analyzing various transportation facilities in both interrupted flow and uninterrupted flow environments. HCS files store the network's layout in an arbitrary XY coordinate system with three georeferencing options: the XY coordinate system, the location on a map, and non-georeferenced. Besides, the HCS Urban Streets adopt an XML-styled (.XUS) file to transfer the network geometry, turning movement volumes, and signal timing plans. Moreover, it supports file instance-level conversion to the CORSIM TRF file. TransModeler can use its information to simulate a signalized intersection and generate a quick animation directly from HCS version 7.7 or later (TransModeler, 2020).

Synchro is a software package for modeling and optimizing traffic signal timing, which adopts encrypted files for storing data. Its database utilities make it possible to transfer network layout and geometry information to a universal traffic data format (UTDF) in column aligned or

comma-delimited text files. The information in the standard CSV files includes layout, lane, phasing, timing, and volume.

The TSIS (the Traffic Software Integrated System) is a software package distributed by FHWA that includes the CORSIM as the microscopic traffic simulation engine and a suite of supporting tools (Mears, 1999). In CORSIM, all input network, traffic signal timing, and volume information are entered into column-aligned text files. Each line in the input file is considered a record and has an integer number indicating what data is stored in that record. All records used to represent a network scenario are ensembles and are ranked in a determined order in a single text file, namely the TRF file. Both Synchro and TransModeler can convert the CORSIM TRF format into its network format.

TransModeler is a traffic simulation package developed by Caliper Corporation, which applies to a wide array of traffic planning and modeling tasks (TransModeler, 2020). Unlike the column-aligned format of CORSIM, it stores the network as a combination of layers in a geographically referenced database. Furthermore, each layer has a unique set of fixed attributes designed to maintain the relationship between various simulation database layers. The layers of a simulation database (.DBD) include links, segments, lanes, nodes, lanes, connectors, centroids, centroid connectors, sensors, signals, and vehicles. Besides the network information in the database, input files in the project level setting also include demand (.TXT), signals (.TMS), Incidents (.INC), Pedestrians (.PED), HOT lanes (.HOT), detour paths (.PTH), turn prohibitions (.BIN), and other inputs designed for transit, routing, and parameter setting. Even though TransModeler has features to import networks from other software, the binary encryption of its input files and database makes secondary development challenging.

VISSIM is a commercial microscopic simulation tool with over 7000 licenses distributed worldwide (Fellendorf & Vortisch, 2010). In a new project of a simulation run, the generated files in the VISSIM package may involve XML-based files such as network files (.INPX), layout file (.LAYX), signal control file (.SIG), volumes, and routing files (.ANMROUTES) files and a Path file (.WEG), and simulation run database (.SDF file) in database format. Furthermore, to help the developer building interface, VISSIM implements Microsoft Component Object Model (COM) as its programming interface. The COM interface provides access to: 1) the modeled road network with all its attributes, 2) signal control, 3) evaluations, 4) all vehicles in the simulation and their attributes, and 5) the simulation control.

SUMO is an open-source traffic simulation package supported by the German Aerospace Center. It is not only a traffic simulation software but a suite of tools that can help prepare and perform the microscopic simulation (Behrisch et al., 2011). The input and parameter files are specified in the XML format except for the Origin and Destination related inputs that can be stored as a text file. Furthermore, it is the first simulation package that officially supports the remote-control interface. Owing to its high degree of openness, it can communicate with external applications and adapt to an online environment during the runtime of a simulation.

2.3.2 Data scheme

Link-Node Diagram

Some software programs use a link-node scheme to represent the road network, where nodes are the intersection of two or more links. Links represent the length of the roadway segment between the nodes and usually contain data about the geometric characteristics of the roadway segment. The two-way links coded by the user are then represented internally (inside the software)

as two one-way links. A node-numbering scheme is a process that facilitates error checking and the aggregation of performance statistics for groups of links related to a specific facility or facility type. It reduces the search for results in the massive text files. (FHWA, 2019) The data used in a link-node scheme typed simulation software usually includes: link geometry, traffic control, traffic operations, management, traffic demand, driver behavior, events, and simulation run control data.

Universal Traffic Data Format

Universal Traffic Data Format (UTDF) is the first standard specification for data transfer between various software packages. It uses text files to store and share data. The UTDF scheme can also share data between software and traffic signal controller hardware and hold multiple volume counts and multiple timing plans for the same intersection (Albeck & Gerken, 2017). Both comma-delimited (*.csv) and column-aligned (*.dat) text files are supported; the conversion capability of the UTDF format is shown in Figure 2.2.

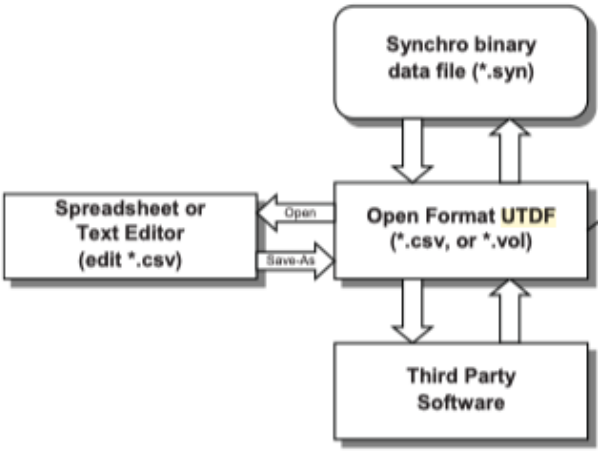


Figure 2.2 The API design of the open format UTDF

AMS Data Hub

The AMS (Analysis, Modeling, and Simulation) Data Hub is a unified database schema developed during an FHWA research project. This data standard aims to improve the integration effectiveness of analysis modeling and simulation tools across various domains and scales. It is designed with five critical goals: 1) Fully documented and available royalty-free; 2) Developed through a publicly visible, vendor-independent process; 3) Implementable by both proprietary and free/open-source software; 4) Interoperability among diverse traffic modeling and simulation tools, and 5) Sufficient and unified in its representation at various spatial and temporal scales. The CSV-based database interconnects with four different data sources as shown in Figure 2.3.

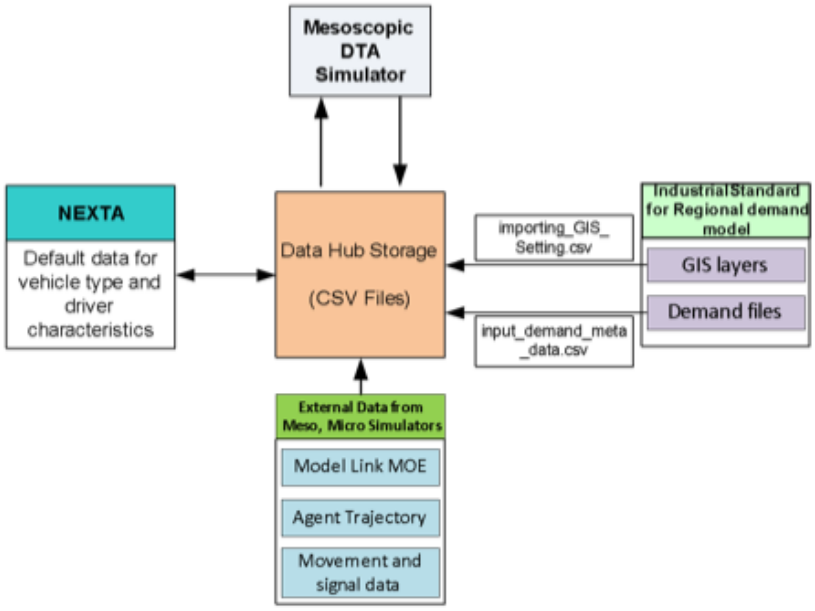


Figure 2.3 The data workflow of the CSV database (X. Zhou & Nevers, 2014)

The CSV-based database contains the following worksheets: Configuration, Node, Link, Zone, Activity Location, Demand Matrix, Middle-income, Speed Sensor Data, Count Sensor Data,

Signal Timing Plan, Movement Data, Phasing Data, etc. The data structure diagram of different data layers is illustrated in Figure 2.4.

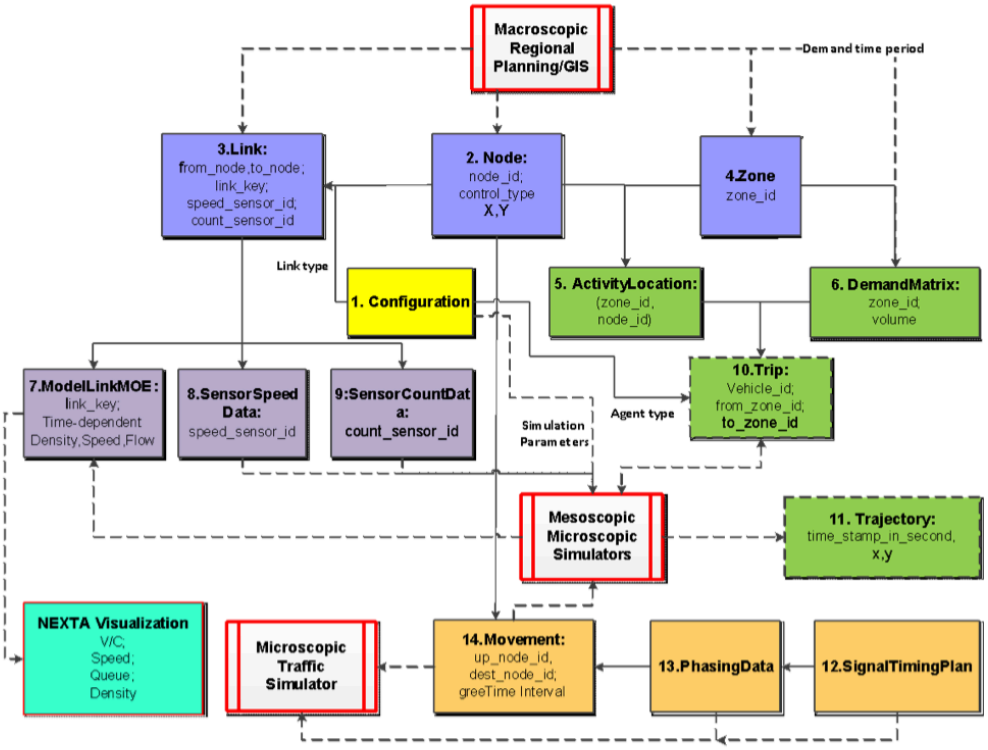


Figure 2.4 The data structure diagram in the AMS data hub

GMNS Data Hub

The General Modeling Network Specification (GMNS) is an open-source specification developed by the Board of Zephyr. It defines a standard human and machine-readable format for sharing routable road network files, intending to represent the network in multi-modal static and dynamic transportation planning and operation models. At a high level, GMNS simply models a network of nodes and links. Meanwhile, users can put in as much detail as adding lanes, movements, geometry information, etc.

2.3.3 Cloud data stores

Version control and asset management are two critical facets in the execution of a complex project. The conventional method of local modeling and data storage is inefficient in updating and coordinating information among team members. Therefore, efforts are continually being made to use online collaboration and management tools using web and internet architecture. To simplify documenting, storing, and locating VISSIM models, the Washington State Transportation Center developed a Simulation Archive (SA) website to guide users through a browser-based interactive tool (Beaulieu, 2007). In comparison, other entities utilize a third-party online code hosting platform to facilitate model management. Some researchers have adopted the GitHub repository, and sourceforge as the cloud storage place. In this case, the crowdsourcing mechanism can collect more qualitative network cases. In the meantime, those networks have better opportunities to be exposed to developers of interest to facilitate research and enhance practice.

Transportation Networks is a GitHub repository maintained by Dr. Hillel Bar-Gera's TNTP from 2016. It was designed for academic research purposes only, especially in the traffic assignment problem. However, the TNTP Data format is tab-delimited text files, with each row terminated by a semicolon, which is not a transferable network format to other platforms. Thus, it is not generic to the practice. Also, since the node information is not geocoded it is incompatible with GIS software, the user case remains stored in the local desktops (Transportation Networks for Research Core Team, 2016/2020). GMNS data adopt the GitHub repository in 2020 to demonstrate the small network example and accumulate conversion tools. The conversion tools are usually written in a platform-independent programming language like R or python, aiming to convert networks from sources like DynusT and OpenStreetMap.

2.4 Conclusion

Considering the enormous manual effort required for simulation tasks such as network coding, data format conversion, error checking, and sample network sharing, the transportation network modeling committee conducted various studies to improve the modeling efficiency, as reported in the literature. While these studies have made significant contributions to data standards and conversion automation, some of the following critical issues still need to be addressed.

1. How to improve the data interoperability from various data sources and file formats within a simulation service.

2. How to consider the tradeoff between software capability and software extensibility for the service experience.

3. How to best leverage cloud storage and cloud computing for a data service in a multi-resolution simulation context.

The remaining chapters will focus on developing a data schema, compiler, simulation function APIs, and a cloud platform for transportation system modeling.

Chapter 3 Research Framework and Platform Structure

3.1. Introduction

This chapter illustrates the overall research framework and the interrelations between its principal components. To improve the data interoperability, the setting of base service logic can accommodate various data service workflows among different stakeholders in the ad hoc

simulation service. The first section introduces a cloud architecture for service providers to integrate core capabilities to function at the back end. The second section focuses on manipulating the simulation function and meeting project requirements, involving a web interface to complete the simulation life cycle. Finally, the third section presents the transportation management scenario in the cloud context to facilitate professional communication among platform end-users.

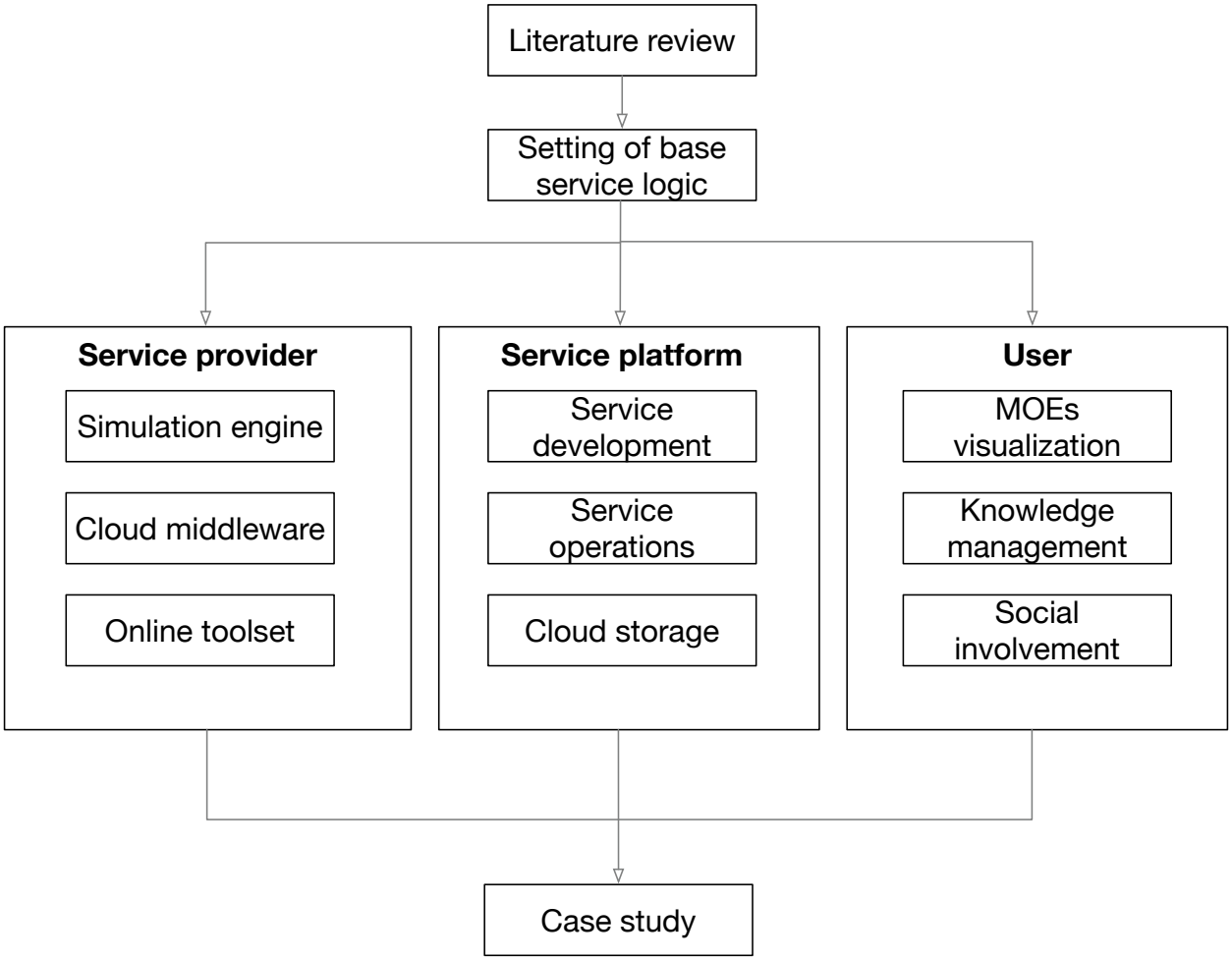


Figure 3.1 The proposed research framework

Figure 3.1 shows the overall flowchart of this research. The first section is the architecture for the simulation service provider with separated modules. It consists of the following three major tasks: developing a general data compiler for the simulation engine, the cloud middleware for data exchange, and developing an online toolset for simulation data collection and processing. At the top of the service provider section is about the data compiler of simulation engines. At the top of the service provider section is the data compiler of simulation engines. In developing the general data compiler for mainstream simulation engines, which's critical task is identifying standard data pipelines. The three main components in the pipeline are data preparation, data storage, and data representation.

In the middle of the service provider section is to develop a cloud middleware for two-way communication. The process contains establishing a stable connection port for the simulation engine, receiving the service request, scheduling the request, conducting the simulation task, and sending back the output files for online post-processing. The last part is the toolset deployed on the cloud platform, a series of custom scripts that the service provider wants to deliver to the user.

The second section in the middle shows tree components in the service platform, which are used to enhance the service experience through a web interface. Similarly, three significant components exist in the platform modeling process. First, the service development is constructed for the service provider. It involves the assembly of the desired online tools and the workflow to put them on the shelf. The demand for externalization of the base service logic entails the development of service operations, which contains the design of user experience and

implementation of the service life cycle. Finally, cloud storage serves as the workspace for both the service provider and the user.

The last section is designed for the simulation deliverables, which consists of three components as well. In terms of the output delivery, it involves the classification, aggregation, interaction, and visualization of the simulation MOEs. Following the output representation, the knowledge management module enables users to share and reuse the developed knowledge in a scalable manner on cloud storage. At last, social involvement entails various instance status management among multiple perspectives the value propositions along the service life cycle.

3.2 Simulation Provider

3.2.1 Key features

To deliver the service to the user, the platform architects a back-end solution for the service provider to deploy their capabilities. Figure 3.2 illustrates the proposed integrated back-end solution between principal components: simulation engines, cloud middleware, and online toolsets. The bottom block depicts the two main modules designed in the supporting platform. The left module, cloud computing service, features the management of virtual containers for the service provider. The right cloud storage service block supports the input and output files of the computation services. The cloud middleware in the middle maintains a two-way communication pipe between the tools on the platform and simulation engines in ad-hoc servers. Note that the core notion of back-end system architecture is to, customize the tools that meet the users' simulation requirements and deploy them online, which requires smooth cloud communication and an operational simulation engine. Any requests ordered by users would be transformed and sent in the knowledge-powered service tool. The service tool would capture the needs and processes and

compile them to an input file that is compatible with the requirements of the simulation engine server. The simulation engine server is configured to take a standard input file, trigger simulation runs, and generate outputs. If the data flow decoupling between the tools and the simulation engine is successful, the service tool can be deployed online. The proposed cloud middleware will then activate the communication tunnel between the tool and engine. The decoupled design of the previously integrated components could significantly improve the utility and reusability of each component.

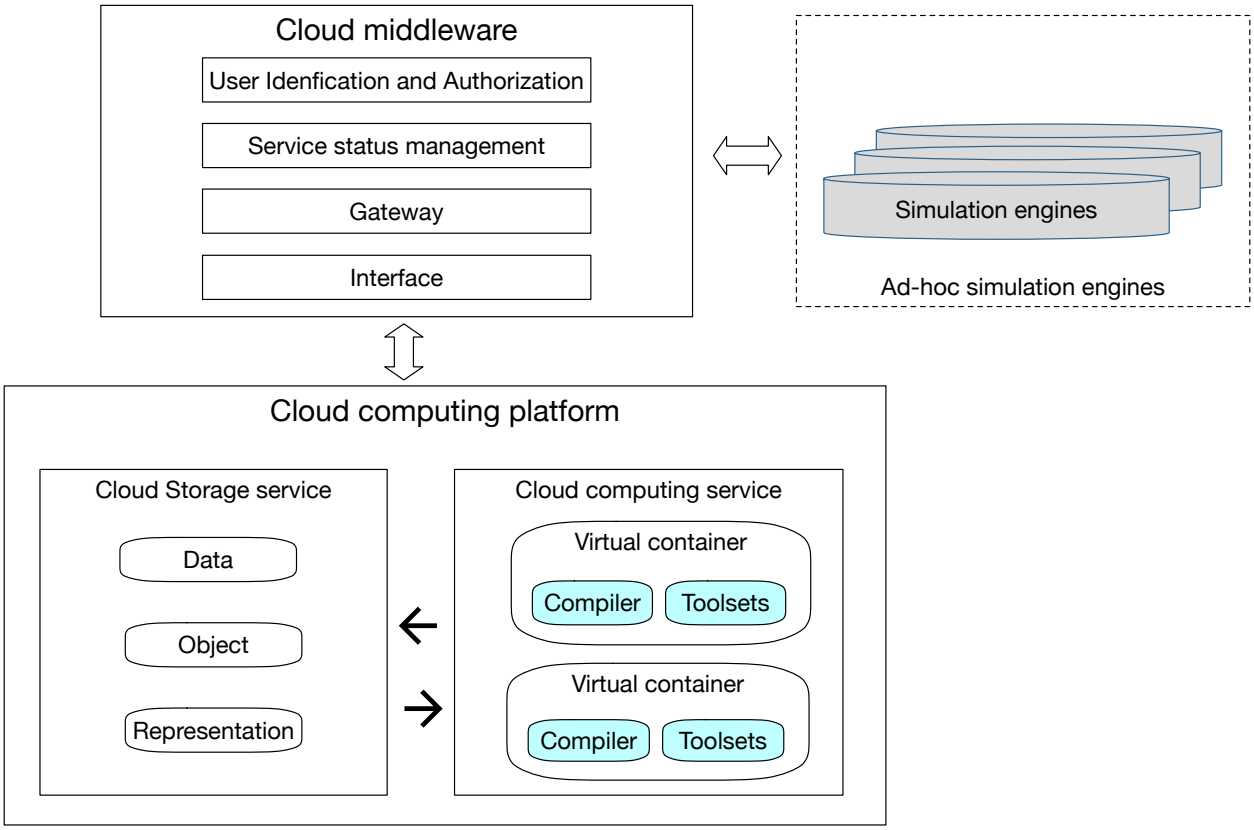


Figure 3.2 The back-end system architecture for simulation provider

3.2.2 Data containers

The data container refers to a type of instance that can take, store or process data. In a simulation service, the data container serves as an intermediate object that facilities the service provider to flexibly interact with the user. The cloud platform providers three kinds of containers, which include:

- **File instance**: covers all the mainstream file types. This representation could encompass serialized data like CSV, structural data such as XML and SHP, or even macro-enabled data formats such as XLSX and XLSM.
- **Data model**: symbolize a structuralized data objects, which could be easily transferred using the format of JavaScript Object Notation (JSON.) It is a visualizer of the user's needs embedded with a preliminary data validation function that supports lean development by combining a series of form components.
- **Program instance**: acts as the prototype of an online service tool. It empowers the structural and data models with high-level functions to convert the raw data into the ideal output. The capabilities may include but are not limited to the following types: validator, pre-processer, converter, visualizer. The combination of file instance, data model, and program instance is a flexible practice that supports data polymorphism (Dean & Dean, 2021).

3.2.3 Primary data pipelines

Suppose data is structured and contained in the container. In that case, it could be packed into a data pipeline for further process. Four primary data pipelines used in the cloud platform are listed below.

File instance + service tool + cloud engine:

Due to the inflexibility of the current subscription scheme, and the limitation of the software license dissemination method, the availability of activated software is impeded. For example, a consultant might hand over an archive and the documentation of a simulation file to a client after completing a project. Suppose the input files are runnable and the process is reasonable, the owner still cannot run the program unless the owner has a valid but expensive license. But what if the owner could send a request to a cloud engine through a service request tool and wait for the simulation outputs. Note that the cloud engine here stands for the combination of cloud middleware and simulation engine.

Service tool + data model + cloud engine:

The current autonomous network modeling tools, such as OSM2GMNS, can convert public data from an open-source map data provider, such as Open Street Map (OSM). This kind of program can be packaged into another service tool named the network tool. However, the consolidated network generated from the autonomous process usually needs extra information to be converted to a standard input format. Hence, the data model plays a critical role in collecting and transferring the structural data object. The role of the data model and its applicability to different components will be described in Chapter 4.

File instance + data model + service tool + cloud engine:

Planning operators often need real-time feedback from a simulation engine to analyze a short-term regulation's potential impact on day-to-day operational transportation management practice. Moreover, short-term regulations only involve routine operations with limited inputs. Therefore, the standard file instance can perform as a base file. Building upon that, combining the data model and service tool creates a customized simulator, which is demonstrated in the case study in Chapter 6.

Data-as-a-Service:

From the perspective of crowdsourcing, when the number of standard file instances reaches a certain amount in the cloud data pool, the Return On Investment (ROI) would reach a tipping point (Mateljan et al., 2010). This is when a stand-alone crowdsourcing data server should be set up, which would build a more seamless data pipeline for more users. In such circumstances in the transportation field, the Simulation-as-a-Service mode could drill down to the Data-as-a-Service mode. The social involvement section in Chapter 5 would discuss the mechanism of the Data-as-a-Service.

3.2.4 Functional requirements

To provide data pipelines for a multi-resolution simulation, this research designed a cloud architecture that has the following functions:

- Receives the geometry, signal, and demand data of the network inputs on the Cloud through the data container.
- Pre-processes the data via three-step procedure of: validation, metamorphism, and compilation, to create the standard input file using the online toolset.
- Activates, schedules, assigns and manages the simulation tasks to the idle simulation engines

through the cloud middleware.

- Receives the outputs from the Cloud and completes the instantiation of the service tool or program instance.

On top of the black-box model of computer systems, which represent the input-model-output structure, Figure 3.3 depicts the SimaaS cloud architecture's core logic. The cloud computing platform covers five black-box functions over the input, model, and output quadrants, whereas the cloud middleware and simulation engine function within the quadrants of model and output. The direction of the dashed arrow stands for the general execution sequence of the black-box functions over the black-box model and across the cloud computing platform, cloud middleware, and simulation engine. Note that the functions in the simulation engine are interrelated and have no definite sequence. A detailed description of each function will be presented in Chapter 4.

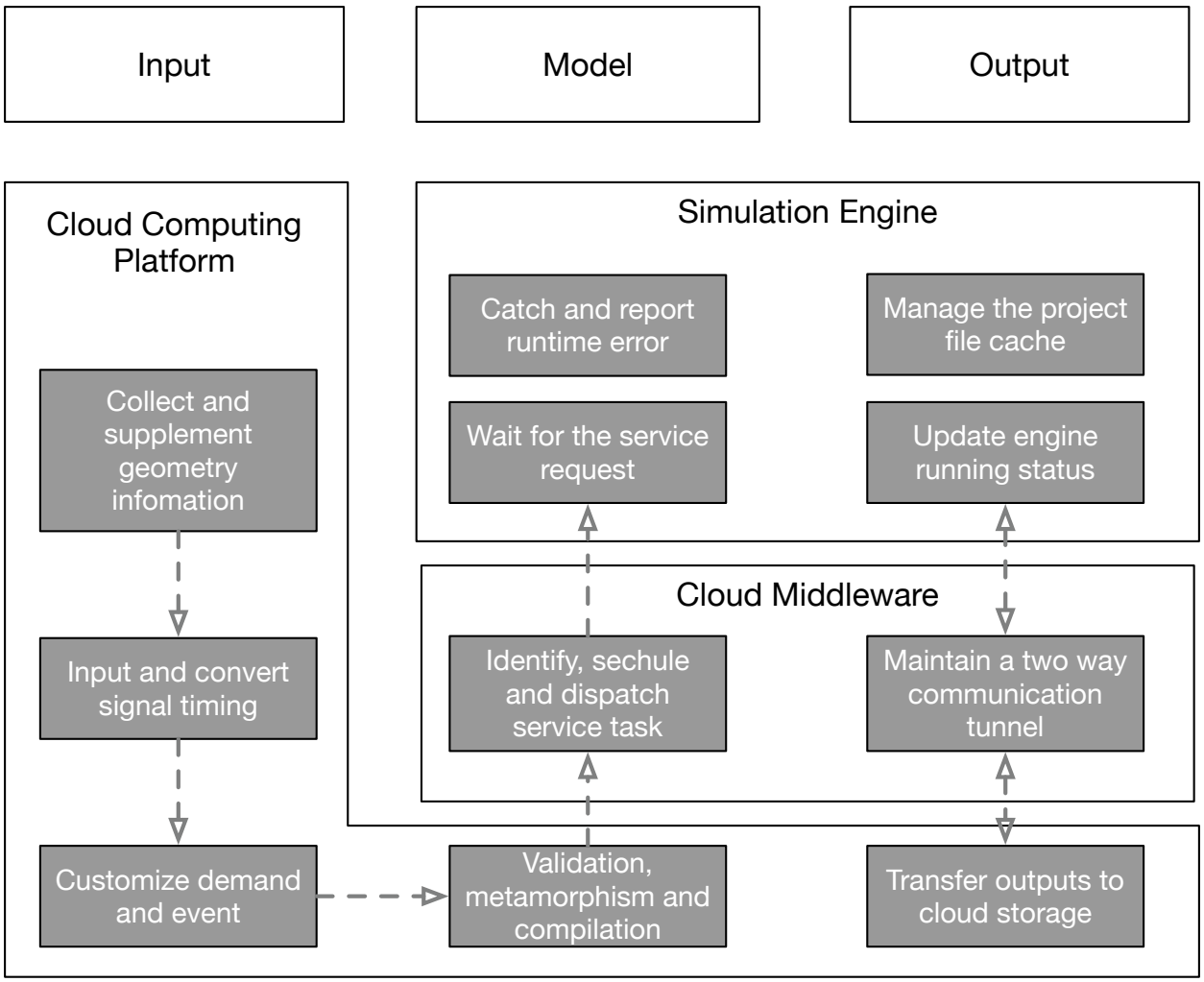


Figure 3.3 Flowchart for the SimaaS cloud control mechanism

3.3 Service Platform

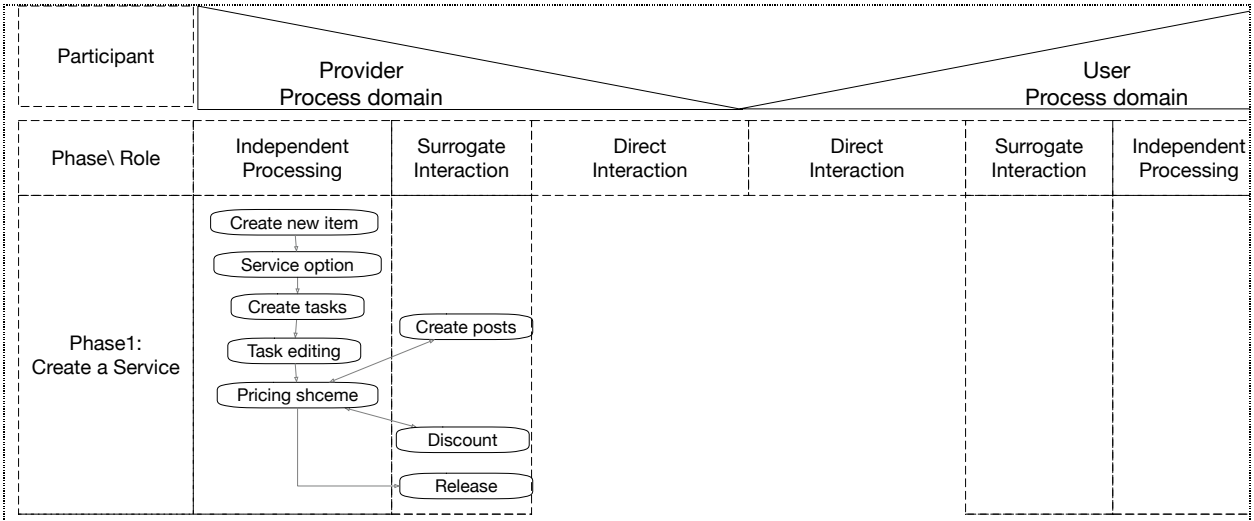
3.3.1 Key features

A service platform is a third-party infrastructure that provides the workspace and necessary interfaces for service providers and users. On the one hand, it enables the service provider to combine and assemble the data pipeline and online toolset into a customized simulator and

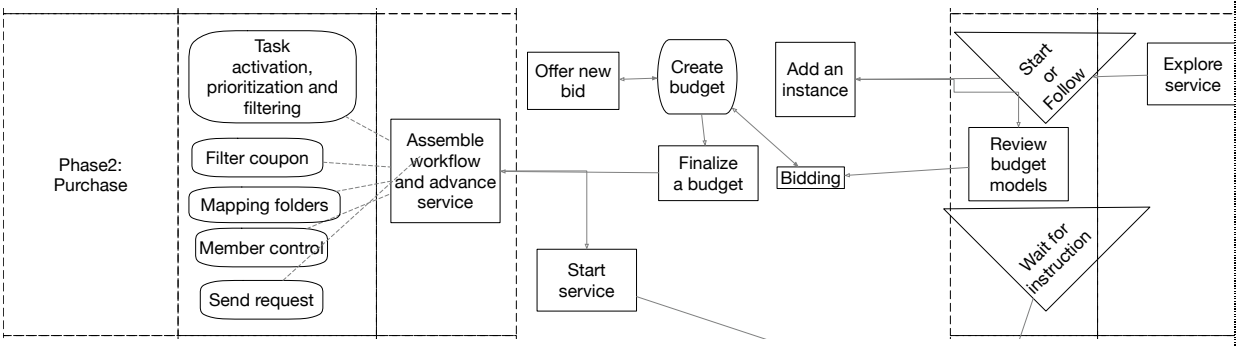
function as a workflow. The workflow can be stored as a service instance after the completion of the service lifecycle: service development, purchase, implementation, and transaction. On the other hand, it provides the user with a direct interaction interface to feed requests, inputs and check the service deliverables. Instead of creating a new simulator, the user can choose from the existing service instances and activate a new workflow from the listing instances in the platform market. Therefore, the simulator could be highly customized and extensible, which is beneficial to users considering the capability and extensibility of the toolset package; the simulator can also be standardized and capable for other users to repeat the workflow.

The base service logic is visualized through a Process-Chain-Network (PCN) Analysis, which is shown in Figure 3.4., where a process chain is simply a sequence of process steps with an identifiable purpose. The arrows represent a state dependency, and the dashed line suggests a loose temporal dependency, which means there might need to be a negotiation before moving to the next instance. The PNC diagram can separate the independent processing action and surrogate interaction, for both process entities, and highlight the direct interaction between two entities. In general, a complete cycle of a simulation service needs goes through four phases to complete the tasks. Figure 3.4.a shows the sequence of tasks when the service provider creates and composes a service. When the service is published, several actions need to take by the service provider to make the service instance visible and purchasable. This purchase stage shows in Figure 3.4.b, suppose a user finds an interesting service. In that case, he or she can reach out to the service provider in the platform and initiate a direction interaction process in terms of the budget. Then, with the confirmation to move forward, Figure 3.4.c illustrates the steps that could happen during the

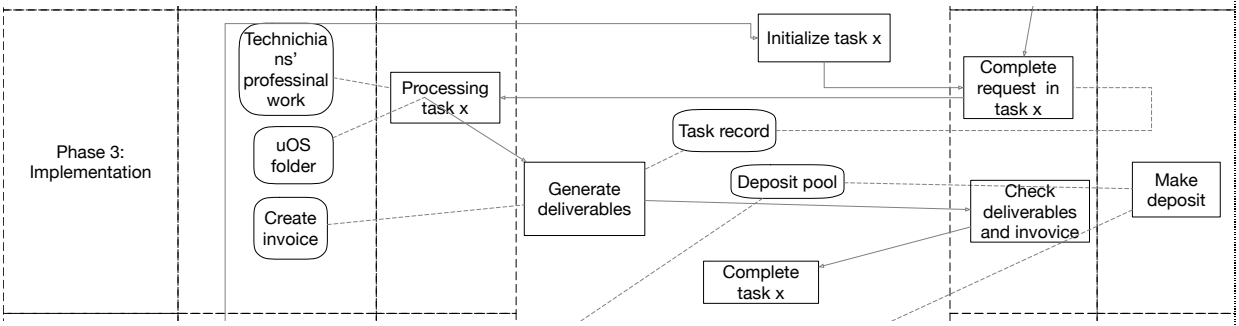
implementation phase. Finally, Figure 3.4.b demonstrates task-based payment steps upon the completion of each task in a workflow.



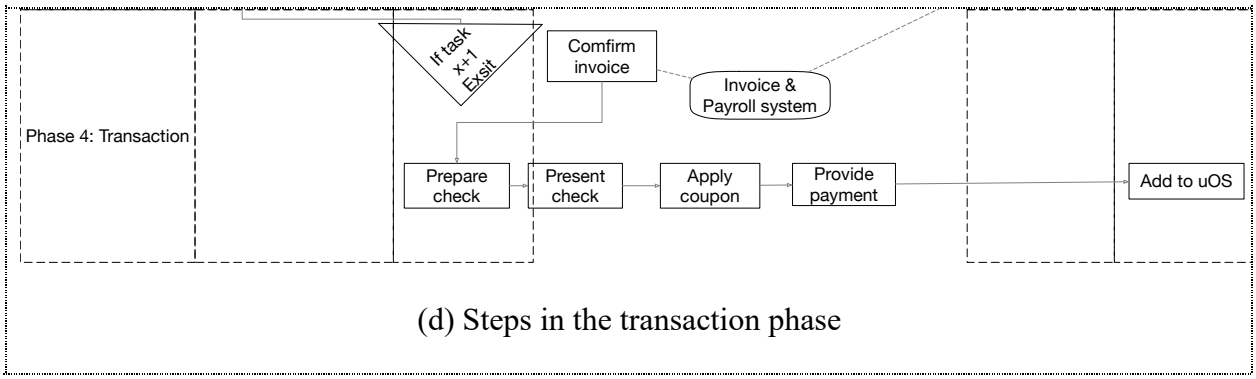
(a) Steps in the service creation



(b) Steps in a purchase operation



(c) Steps in the implementation phase



(d) Steps in the transaction phase

Figure 3.4 Process-Chain-Network Analysis for the base service logic

3.3.2 Primary components

A desktop refers to a workspace for users or providers to find, use, and organize data containers and perform service interactions. The desktops that can operate the above interactions are the primary components in the platform section, which consists of the following:

- **Folder**: inherits the features of the tree structure that are compatible with any operating system. Users can upload, open, delete, move, and apply functions to the data containers on the top of the tree structure.
- **Picture**: represents a flattened organization of data containers by taking advantage of the information on pictures. The data containers could be bound to any representative objects in the picture
- **Map**: symbolizes an extensible flattened organization of data containers. It supports layer control to manage various Geographical JavaScript Object Notation (GEOJSON) objects on the map. Each GEOJSON object could represent a file instance, data model, program instance,

and even folder.

3.3.3 Model-Driven development

The approach used to design the service platform is Model-Driven Development, which is oriented to devise conceptual models from different points of view, levels of abstraction, and granularity. Domain ontology could be abstracted from the context model of the user, environment, and platform. Based on the separated ontologies, the service process model as shown in Figure 3.5 and interaction should be modeled in the first place. Then, a platform-independent interaction model (PIM) is a goal when the service process model reaches a consensus. A PIM is a model with a high level of abstraction, independent of any technology or implementation language, that exhibits a sufficient degree of platform independence to allow mapping to one or more platforms. Next, the developer and product manager can use a specific Unified Modeling Language (UML) diagram to describe the user interaction for the proposed platform. Last, the software developer will take full advantage of all the interaction and ontology details to transform service context into source code.

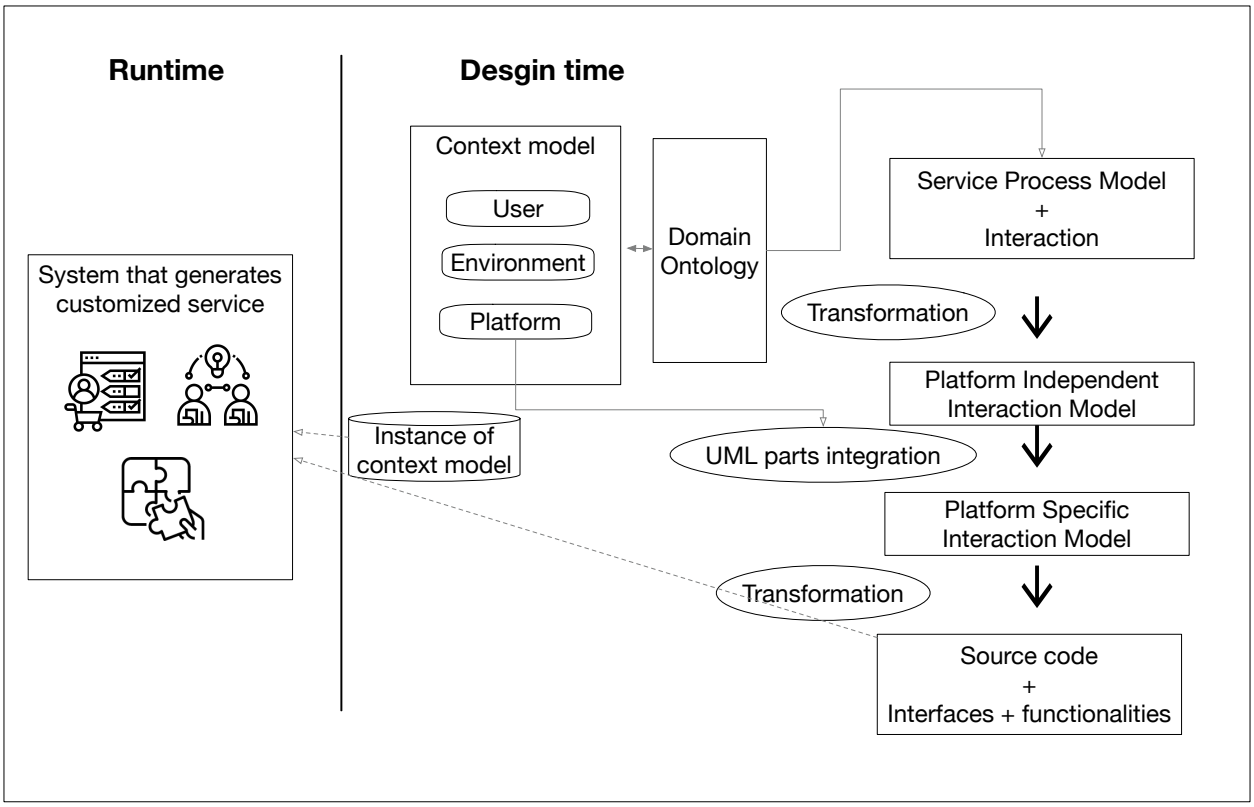


Figure 3.5 The Model-Driven design of the platform

3.4 User and Service Experience

3.4.1 Key features

Every service needs to ensure service provider and user are on the same page regarding the key steps in a workflow. An operation desktop is a place on the cloud platform to exchange information and present deliveries. Figure 3.6 illustrates the three-phase service road map for three primary service forms on the platform. The gray to white spectrum on the left side represents the manual effort's involvement in professional assistance, where dark gray represents a high degree of human interaction and white represents a high degree of standardization and automation. The

three modes could be classified as: pure service, mixed service, and quasi-manufacturing. Except for the professional assistance, all steps could be executed and examined on the operating desktop presented by the cloud service platform. Note that a simulation project could be a mixture of several small services with various service types. This mechanism ensures service flexibility, which is compatible with the current consultant delivery mode. Besides, this workflow would improve the existing service efficiency by including more online service tools to improve the degree of automation and standardization. Moreover, the entire workflow in the project performs on the operation desktop, which could be shared, viewed, and inherited by other interested users. In other words, each project could be packaged as a component-based online simulator product.

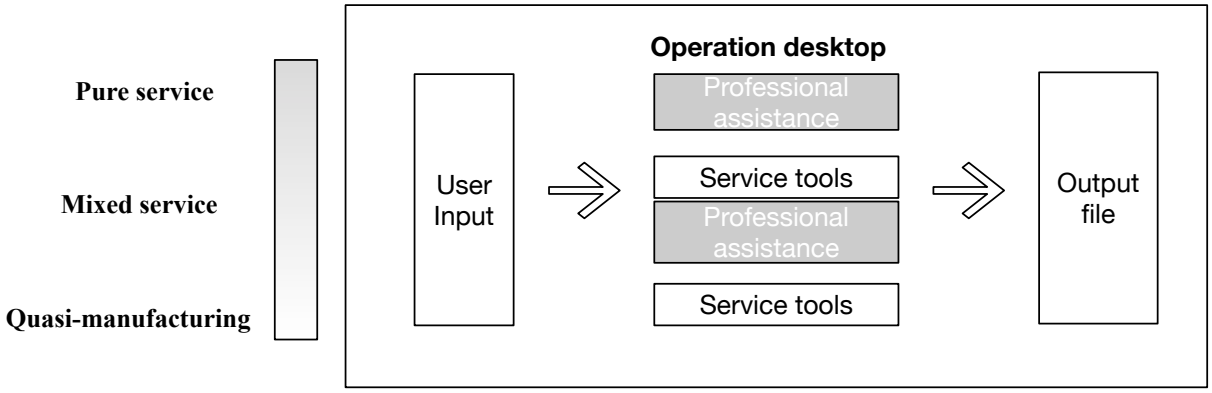


Figure 3.6 Service pattern in the operation desktop

3.4.2 Primary components

From the user perspective, the core behaviors that need improvements are: providing input, examining the output, and sharing the experience. The three following modules are developed to meet those requirements:

- **Network editor**: represents a lightweight data input mechanism. Unlike the conventional and

all-inclusive network editor in the local software, this editor is designed for quick adaption and cloud collaboration to accomplish transportation management tasks.

- **Output management**: covers the mechanism design between cloud storage and service tools. The output files, such as intersection-level signal evaluation and arterial-level control strategy, could be visualized from multiple perspectives through the service toolset.
- **Service hub**: collects and organizes the simulation deliverables for both users and the simulation provider after completing the simulation task or project. The service hub captures the social involvement features on top of the publish/subscribe communication pattern. The users on the service platform can view, comment, like, and share the published content.

3.4.3 Functional requirements

The service should provide the following functions to the simulation user to account for both service customizability and efficiency.

- Customize the simulation scenario and control strategies with the assistance of the service provider on the Cloud.
- Open, input, and run the simulation functions in the customized cloud simulator either individually or collaboratively.
- Receive, examine, and validate the simulation deliverables on the operation desktop.
- Share the service experience and knowledge from the finished work to accumulate community intelligence.
- Inherit the workflow from the published simulation service project to enhance the reusability.

Chapter 4 Component-based Online Simulator

This chapter presents the design and mechanism of an online simulator for performing the critical steps in the transportation simulation. It is expected that a component-based online simulator can identify the potential needs of different user groups and provide them with different functionalities. Also, to address the real-time issue, the core components in the simulator consists of a network editor, simulation engine, error handling mechanism, cloud middleware, and map-based tool.

4.1 Cloud-based Network Editor

Instead of creating an all-purpose network editor interface that integrated all functions in a complicated user interface and user experience logic, the cloud-based network editor is a scenario-oriented, reusable, and extensible interface that supports online coordination. Figure 4.1 illustrates the cloud-based network editor's framework, showing the interrelation between proposed functions and data sources. The left rectangle lists the main application scenarios from microscopic to macroscopic. The right-side cloud-based network editor, a flexible combination of proposed functions and data sources, is assembled for the specific scenario.

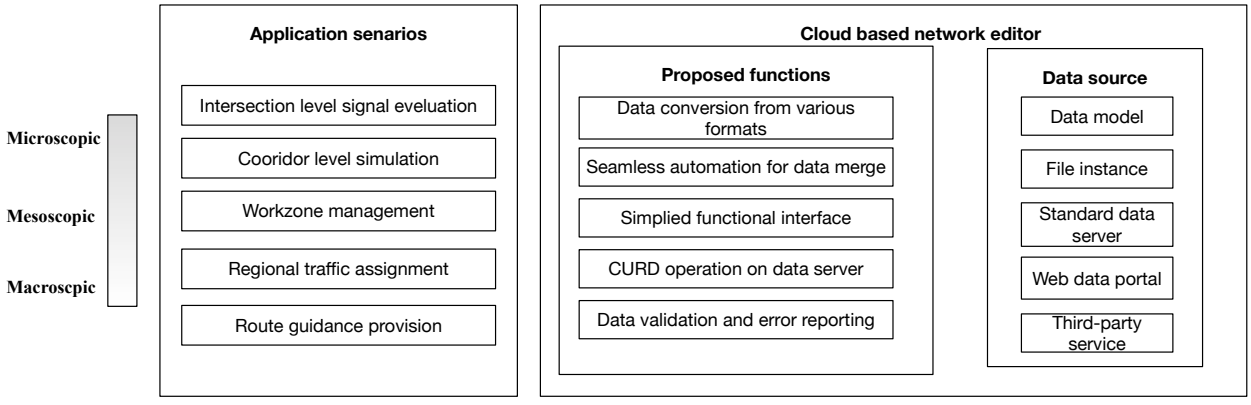


Figure 4.1 The cloud-based network editor's framework

The minimum viable network editor is a program instance embedded with a critical function such as data conversion and could take in a file instance and generate its output. More specifically, the embedded function opens a well-encapsulated interface to connect the user input and output file. On top of that, a scenario-oriented network editor could be a collection of required program instances, a picture mark-up editor, and a map navigated editor.

4.1.1 Data granularity and classification

This study proposes a ternary classification method of the simulation data input to help define the service type in the future. A small experiment is conducted to demonstrate the features of six simulation packages according to the position in this classification. Figure 4.2 shows the classification of the simulation packages and their size. Notably, the size of the dot on the plot represents the number of fields reserved for the simulation software. All the input fields were collected and classified into three main categories: network geometry (Geometry), travel demand (Demand), and traffic control (Control) based on engineering judgment. The position of each point depends on the relative percentage of the three main aspects. Five simulation software were evaluated, including five microscopic packages represented with blue dots and one mesoscopic software represented by an orange dot.

Based upon the schematic representation of three major building blocks for the input of a simulator (Fellendorf & Vortisch, 2010), the classification criteria of three main aspects have been summarized in the following points.

1) Geometry stands for the parameters that affect the network supply capability. For example, it covers the essential link and node geometric attributes, lane alignment information, bus stop location, parking lot, detector, etc.

2) Demand represents the parameters and events that influence the traffic volume and traveler decision. It could be the entry volume at the entry link, the turning fraction at each movement, incidents, lane block, and modeling parameters that depict the traveler behavior such as car-following, lane-changing, and driver acceleration model, and so on.

3) Control denotes the management strategies that the traffic operation department could exert. It includes the signalized and un-signalized control for intersection, the ramp metering for the freeway, the schedule for transit, the attribute for toll plazas, etc.

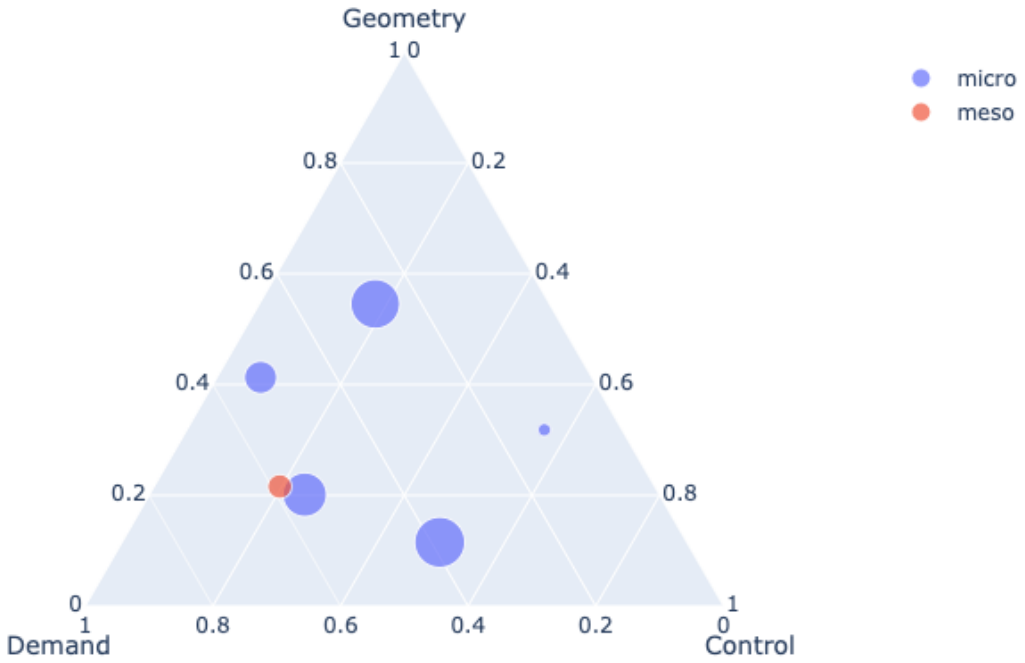


Figure 4.2 Simulation input composition classification

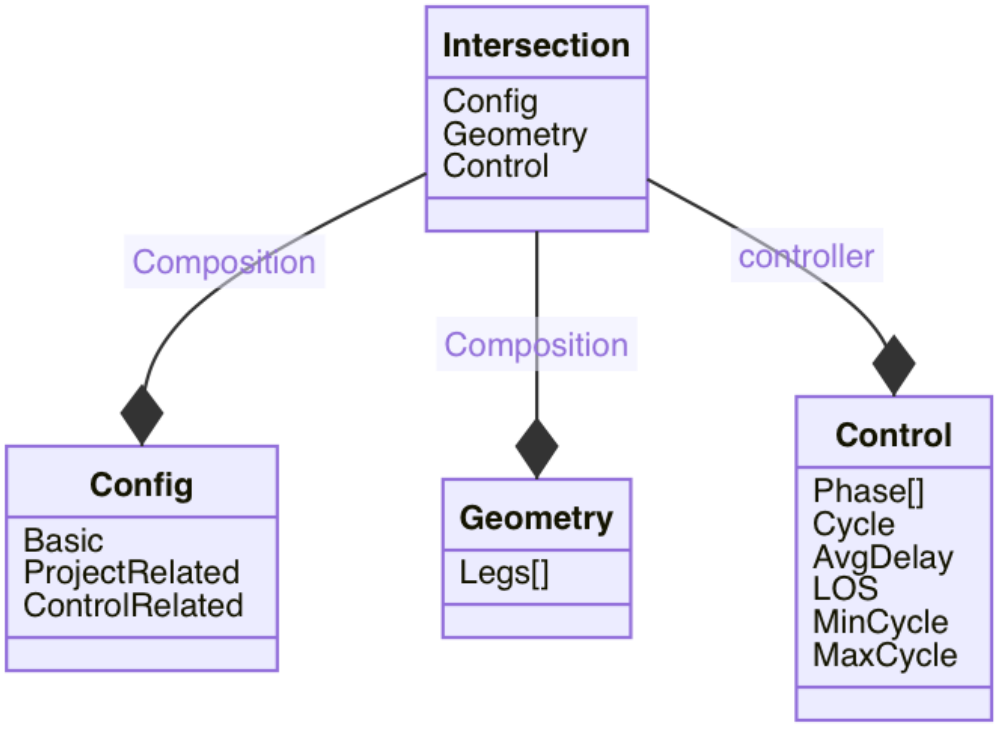
In the input classification experiment conducted, TSIS 5.1 takes 1163 parameters to configure a simulation. Parameters of geometry and demand take 38% and 12%, respectively, whereas the control-related parameters account for about 50%. SimTraffic is a simulation extension based on Synchro 7, which requires a minimum number of inputs. Among the 80 available entries, the control-related input takes as high as 56%. Sumo 1.8.0 is an open-source microscopic and continuous traffic simulation package designed to handle large networks. Based on the documentation, the number of configurable inputs is around 500; the number of control-related fields only accounts for 7%. In other words, SUMO is good at simulating the vehicle dynamics but poor at signal control. The mesoscopic DTAlite 2016 and microscopic TranModeler 5.0 sit closely in this classification, which stands for both packages that can capture the complexity of demand modeling (58% and 56%). On the other hand, since a microscopic analysis focuses on traffic at the vehicle level, the number of parameters needed by TransMoleder is almost four times larger than DTAlite.

This classification method further is evidence that SUMO is usually not used by traffic engineers to evaluate real-life intersections since its network model is relatively coarse compared to commercial applications as Vissim (Behrisch et al., 2011).

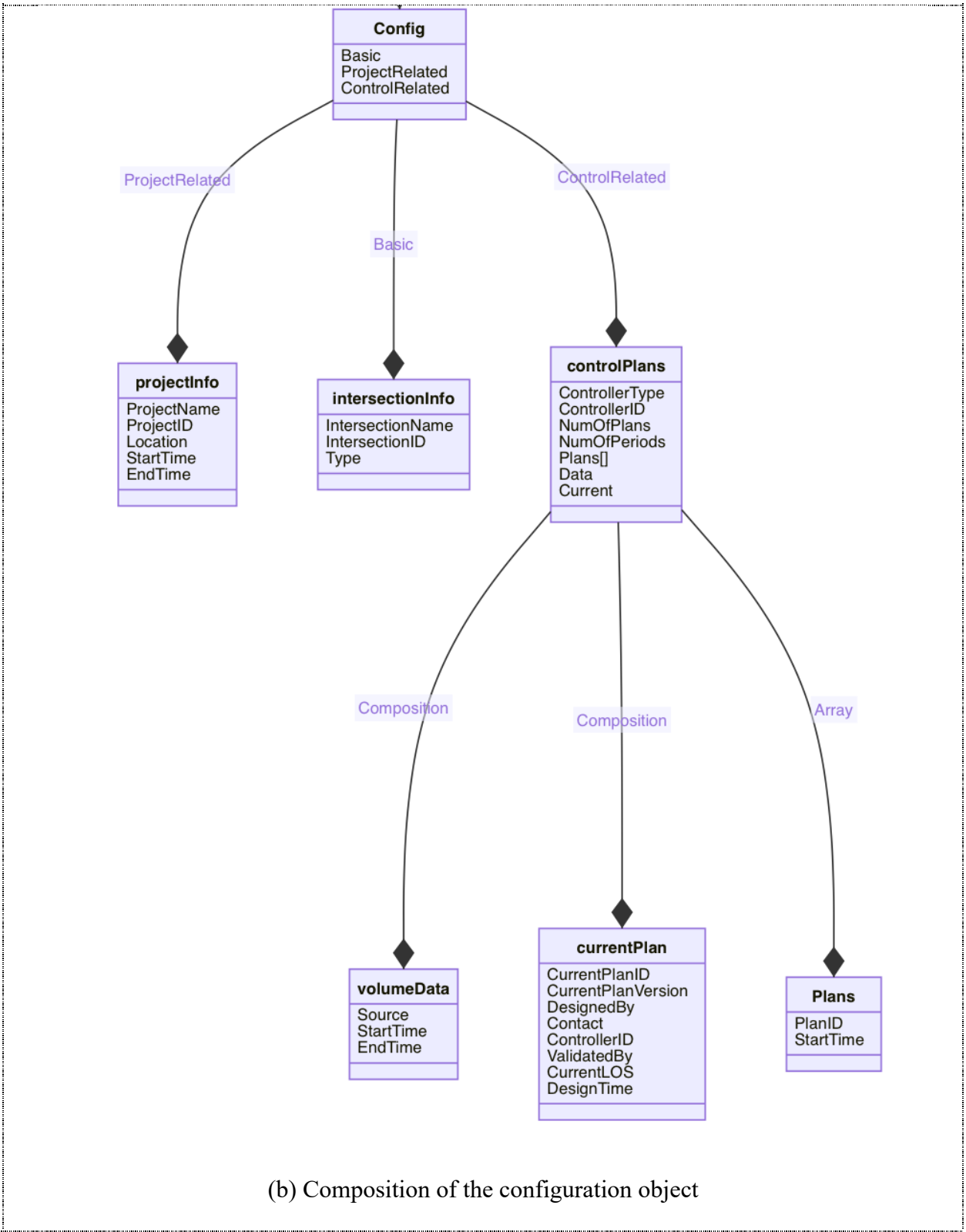
4.1.2 Data compiler for intersection level simulation

This study developed a standard JSON schema for general intersection modeling. Figure 4.3 is the Unified Modeling Language (UML) representation of the data schema that illustrates the composition of a typical intersection object. In this hierarchical representation, the uppermost entities include project configuration, geometry information, and control plan information.

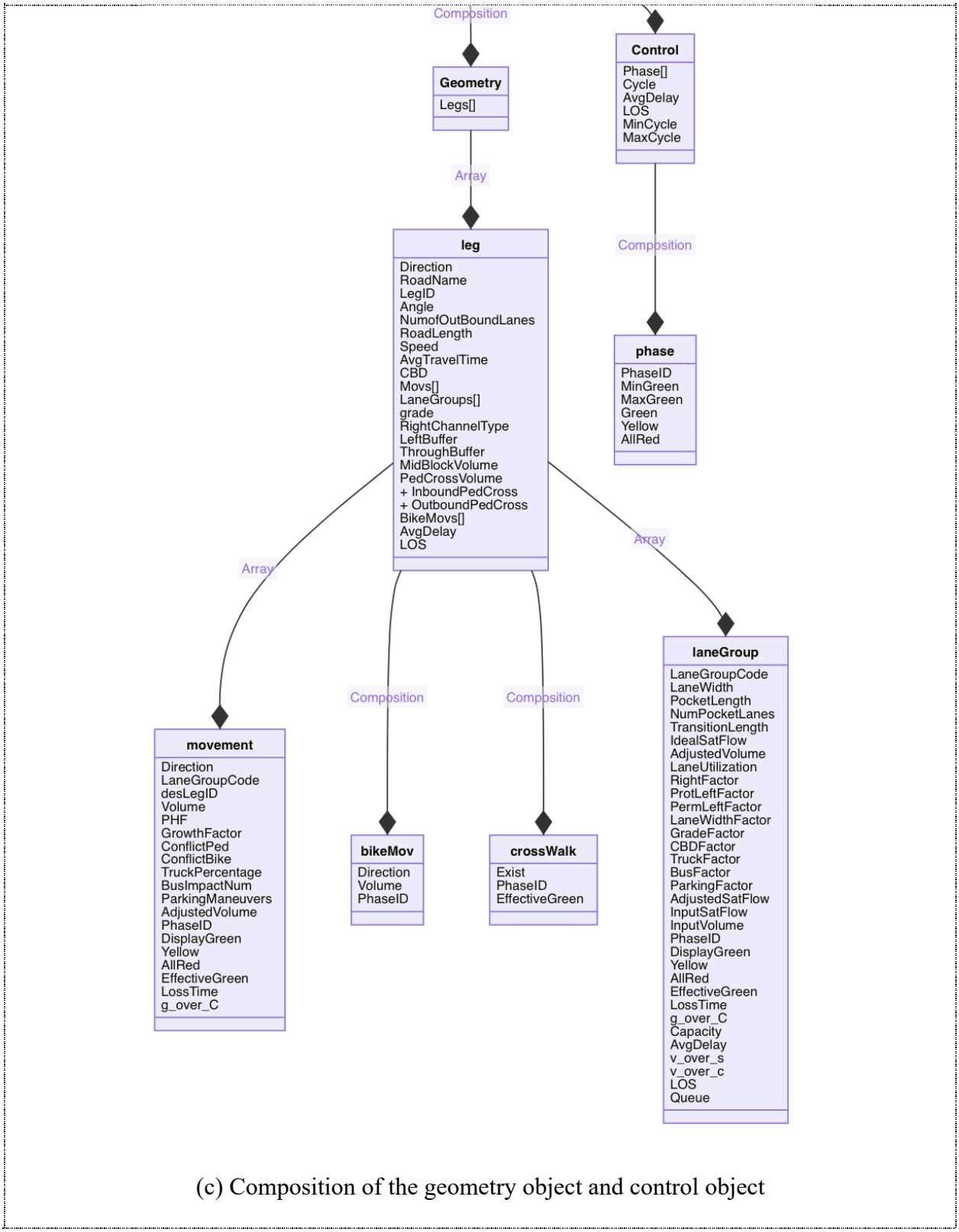
Demand-related attributes such as volume, PHF, pedestrian cross volume, midblock volume, CBD factor are integrated with geometric objects like leg, movement, and lane group. This schema shows the dependency of a leg object in geometric-related information, which consists of two critical, independent object arrays—movements and lane group. By leverage the developed file instance and data model, any intersection could be modeled and parsed into the JSON schema; thus, program instances can work on the standardized information to facilitate the data flow of a simulation service.



(a) General composition of an intersection



(b) Composition of the configuration object



(c) Composition of the geometry object and control object

Figure 4.3 Data scheme of an intersection

A compiler comprises an analysis module to convert a source file into an intermediate object and a synthesis module to convert the intermediate object to the output file compatible with a specific simulation. Figure 4.4 shows the analysis algorithm to convert the source input-(the Excel file instance) into a standard intermediate—the intersection object. Figure 4.5 demonstrates the synthesis algorithm that convert a JSON intersection object into an input format (fixed-column aligned file used in simulation engine CORSIM). It is an example that completes the synthesis step from the intermediate structural object to the engine readable input file, TRF file, in this case. However, to get the intersection information from the user, pre-process, validate and convert them into the intermediate structural object, an analysis step is developed in Excel.

Initialize: Specify a working directory and load the file instance (input file)

Step 1: Define the input directory and name of the file

Step 2: Read workbook data and names of worksheet

Step 3: Initiate object--Configuratioin

    Read and instantiate Project Information from work sheet:Basic Info

    Read and instantiate Intersection Information from work sheet:Basic Info

    Read and instantiate Control Plan from work sheet:Basic Info

Step 4: Initiate object—Geometry

    Get number of valid legs ***num_legs*** from work sheet: Geometry & Volume

    For ***i*** in ***num_legs*** :

        leg[i] = instantiate leg object from work sheet: Geometry & Volume

        For ***j*** in ***movement_range*** :

            mov[i] = instantiate movement object from work sheet: Geometry & Volume

        For ***k*** in ***num_lanegroup*** :

            If lane group classification criteria satisifed:

                lng[k] = instantiate movement object from work sheet: Geometry & Volume

    append list ***mov*** and ***lng*** as the attribute of object ***leg[i]***

Step 5: Initiate object— Controller

    For ***i*** in ***num_phases*** :

        phase[i] = instantiate phase object from work sheet: Phases & Timing

        add ***phase[i]*** object as the attribute in Controller

    Read and instantiate other attribute Controller from work sheet:Basic Info

Step 6: Add objects: Configuratioin, Geometry, Controller together as a integrated object--intersection

Figure 4.4 The analysis algorithm of parsing a file instance

Initialize: Specify a working directory and load the file instance (input file)

Step 1: Define the input directory and name of the file

Step 2: Parse the file instance into **an intersection object--*signode***

Step 3: Create a new blank file and initiate a sequence list ***SL***

Step 4: iterate the sequence list ***SL*** and write parsed ***signode*** attribute

For ***record*** in ***SL:***

    if ***record*** == 1: write project Run Time Infomation

    elif ***record*** == 2: write project Run Control Parameter

    elif ***record*** == 3: write project Time Period Specification

    elif ***record*** == 4: write project Time Intervals and Time Steps per Time Period

    elif ***record*** == 5: write project Reports parameter

    elif ***record*** == 170: write Sub-network Delimiter and Node Coordinate Data

    elif ***record*** == 210: write Time Period Delimiter

    else:

        for ***leg*** in ***signode.legs:***

            en_num = entry_link_number_generator(***leg***)

            target_link = target_link_number_generator(***signode, leg***)

            if ***record*** == 10: write Link Name of the ***target_link***

            elif ***record*** == 11: write Link Description of the ***target_link***

            elif ***record*** == 21: write Surface Street Turn Movements of the ***target_link***

            elif ***record*** == 35 or 36: write Sign or Pre-timed Signal Control Timing

            elif ***record*** == 50: write Entry Link Volumes using ***target_link*** and ***en_num***

            if ***record*** == 195: write Node Coordinate Data of the ***target_link***

Step 4: close file instance and save as the oupfile

Figure 4.5 The synthesis algorithm of converting an intersection object

4.1.3 CRUD operation on the platform interface

The above-mentioned compiler algorithms are about read a file instance and create a new object. To fulfill the complete functionality of CRUD (CREATE, READ, UPDATE and DELETE) operations, the design of the READ and UPDATE mechanism is a critical task for the data operation on the platform.

The mainstream software design uses a data server to accomplish the operations via the server end API. As integrated applications become more complex, maintaining the classic separation of concerns between user interface, data store, and interactional functions come at an increasingly painful cost. This section introduces a lightweight solution to implement CRUD operation on the web interface, which does not involve the database server.

Classical front-end and back-end software architecture

JavaScript is a programming language inherently supported by a browser engine. It can be used to add, delete, and modify components of the Document Object Model (DOM) in an HTML file. Among the new features of HTML 5, one significant improvement is that JavaScript has the FileReaderSync interface to read File or Blob objects synchronously. It allows a script to read data from a file, store data in the browser memory, and download the edited data as a file to the client-side through the browser. However, the JavaScript interface alone cannot write back to the file system in the server side in the cloud context. The only way to create a file on the server is to have codes on the server-side.

Many Web API features now are implemented through asynchronous code, especially those that access or fetch some kind of resource from an external device, such as fetching a file from the network, accessing a database, and returning data from it. In order to function, the server-side code requires a hosted server together with a reachable IP address. The server-side code could be Python, Java, or C++, but the server-side host symbolizes that the right of combination and assemble designed function is at the service provider side. The mainstream front-end and back-end software architecture or the client-server relationship is not applicable also because of the expensive cost for server long time operation compared to the limited use cases.

Session-based front-end and back-end software architecture

In the account of balancing the software capacity and extensibility, a session-based front-end and back-end software architecture is proposed. The back-end code could be purchased and executed by users in the form of toolsets in the front-end of the cloud computing platform. Figure 4.6 depicts the READ and UPDATE mechanism of the proposed architecture. The computing platform is hosted on a cluster of servers, which is a front-end web interface that interacts with the user to realize the tasks in the computing service and storage section. When the computing service section functions are triggered at the client-side, the cloud platform would post an HTTP request on the configured server and execute the requested function with inputs at the back end. As soon as the subprocess is completed on the server, the cloud computing platform would capture the generated outputs and send them back to the cloud storage. In the end, the user-session would be killed when users exit the current editor or log out from the cloud platform.

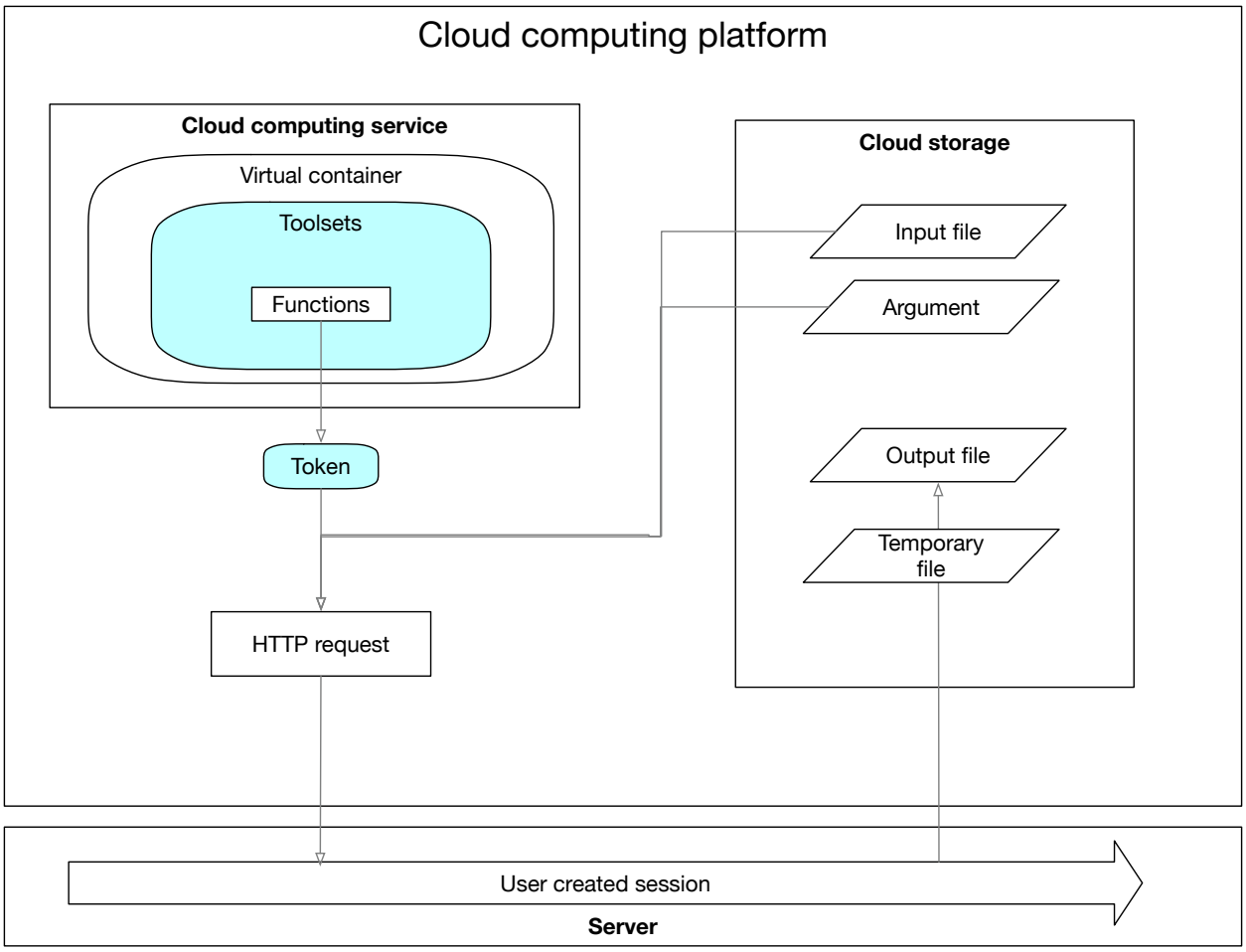


Figure 4.6 Session-based Server-side Function Execution Mechanism

The following shows the step-by-step procedure for a typical cloud-based editor to implement a READ and UPDATE operation in the cloud computing platform environment. Here, the client-side programing language is JavaScript and HTML is the backbone of the editor. In the meanwhile, the platform takes python as the backend language.

Initialize: Specify an operating desktop (folder, map, picture)

Create Session with Token ID

    Use JavaScript to post a HTTP request

Specify a Python function among purchased tools

Attach the uploaded input files

Serialize the Arguments

    Wait for the feedback from server side

    Store the output file such as csv or json in the cloud storage

Parse json or csv file in JavaScript

Present the data in a pre-set structure via HTML template

Manipulate data via JavaScript asynchronously

Confirm and save results

End Session

Compared with the traditional M-1 client-server communication, this architecture leverages the session management functionality of the cloud computing platform to flexibly create a temporary client-server communication pipeline to significantly reduce the server idling time and maximize the usage of its capacity. In the case of multiple users running different tools simultaneously, this architecture can handle an M-M client-server communication, in which the latter M stands for multiple sessions on the server. On the other side, the cloud computing platform also supports the instantialization of every published tool, which means users could purchase the

service as they need. This fine-grained instantialization mechanism could support a much flexible payment plan such as Pay as you Go. From the service provider's perspective, the session-based design helps them reduce the burden of maintaining a server and manage the backend sessions. In this sense, the service provider does not need to own a server; when the back-end code is published, users can just Run as they Need.

Note that the entire data pipeline of a cloud-based editor cannot be accomplished without a bunch of extensible APIs. The study has adopted the following API methods to build the entire stack of the proposed editor:

- Browser environment API:
    - **JavaScript Build-in APIs:** include APIs for DOM manipulating, data fetching from the server, graphic drawing, media playing, and client-side storage.
    - **Third-Party APIs:** comprise the APIs from map providers, APIs for charts, APIs for formatting and table manipulation, etc.
- Platform environment API:
    - **API for the file system:** create, delete, upload a file to the server-side
    - **API for program instance:** collect the arguments and files on cloud storage, execute the function in a token-based session at the backend, send the output to the cloud system
    - **API for temporary massage:** provide a cache space for the JavaScript running environment, support the quick-response function for an editor, and serve as a glue to non-file data transfer.

4.2 Simulation Engine

The simulation engine is a significant component in the Simulation-as-a-Service lifecycle. With the user input collected in the network editor, such information needs to be compiled into the standard input format that is compatible with the specific simulation engine. The simulation engine requires listening to the service request, processing the job, and providing constant feedback during the process. Figure 4.7 illustrates the setup for a simulation server.

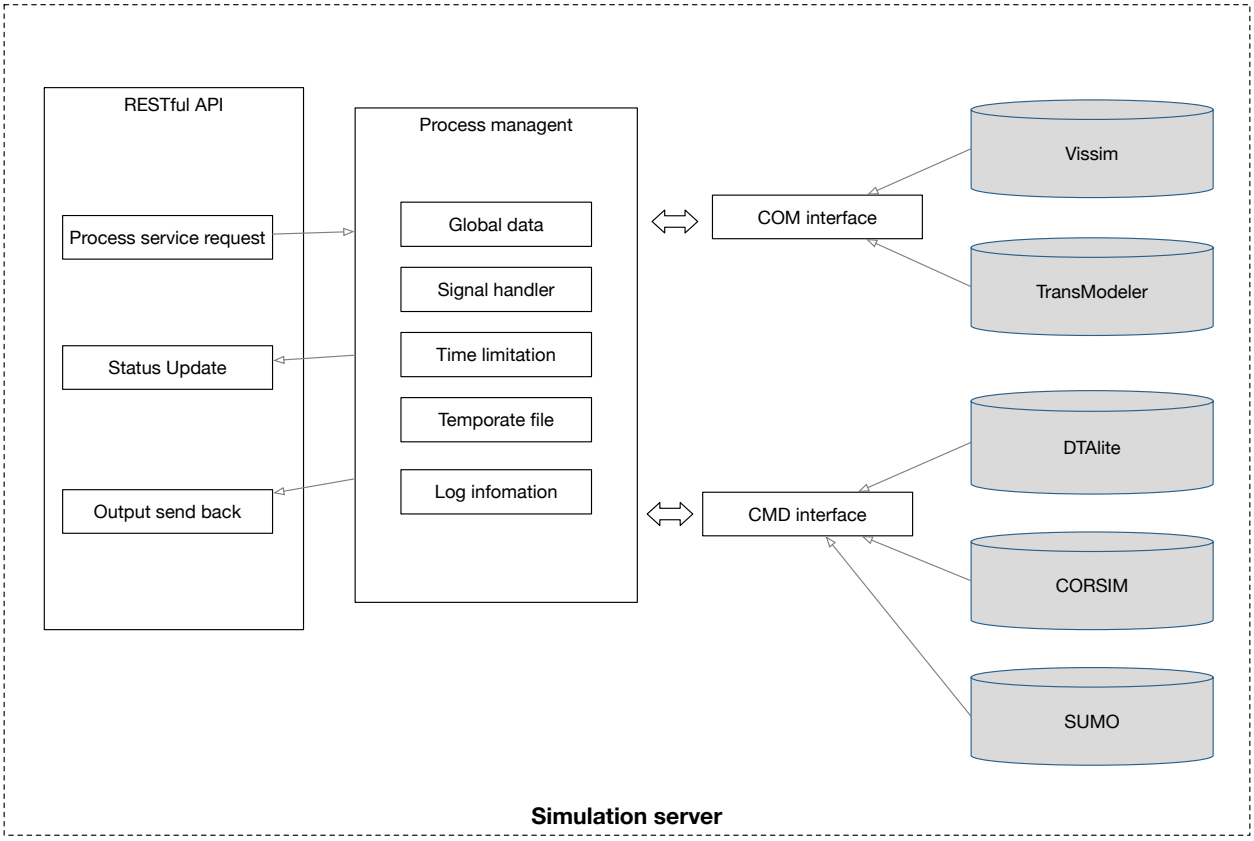


Figure 4.7 The RESTful API architecture and functional design

The general setup of a simulation server consists of three key modules. First, the server must have simulation engines installed as the grey cylinders depict. Since most of the simulation

engines adopt the .NET framework and are only compatible with the Windows platform, excerpt for SUMO, the operating system is recommended to be Windows. Second, a RESTful API service needs to be hosted to receive the request, create the process, provide feedback. Third, a process management module is a critical program that determines the time and method that a simulation engine would function.

There are several vital components in the process management module to ensure the execution of a simulation process. Since the simulation task is an expensive computation process, a Windows server will get overwhelmed if many processes execute simultaneously; the process management module has been developed with the following features to handle multiple requests.

- **Global data:** consists of several variables to represent the status of the process.
    - **Current process:** an instance of the Popen class in Python used to handle the underlying process creation and management. Practical methods include poll (check if child process has terminated), wait (terminate the process after timeout seconds), kill (kill the child), PID (check the process ID of the child process).
    - **Signal handler status:** a Boolean variable that indicates the child process could be terminated or not. Ture means busy; termination is in process, whereas False stands for not busy.
    - **Origin signal handlers:** a collection of signal handler instances. Note that Python signal handlers are always executed in the main Python thread of the interpreter. The default signal numbers include SIGUP (Hang-up detected on controlling terminal or death of controlling process), SIGINT (raise Keyboard Interrupt), SIGTERM (Termination signal).

- **Signal handler:** a function developed to kill the ongoing process at needs according to the signal number received
- **Time limitation:** the upper limit for the simulation process that used to manage the simulation tasks
- **Temperate file and log information:** the error and direct output mechanism regarding the underlying process

With the key components stated, the flowchart of a CMD (Command Line) or COM (Component Object Model) interface-based process could be described in Figure 4.8. Execution information such as memory constraints, debug options, output, error export methods, and time limit could be customized prior to executing a CMD or COM process. A current process instance would be created upon the execution and stored as the global variable for other functions to check its status. If the running time exceeds the limitation, the process would be automatically terminated, whereas the signal handlers in the origin signal handlers' collection would be used sequentially to kill the completed process.

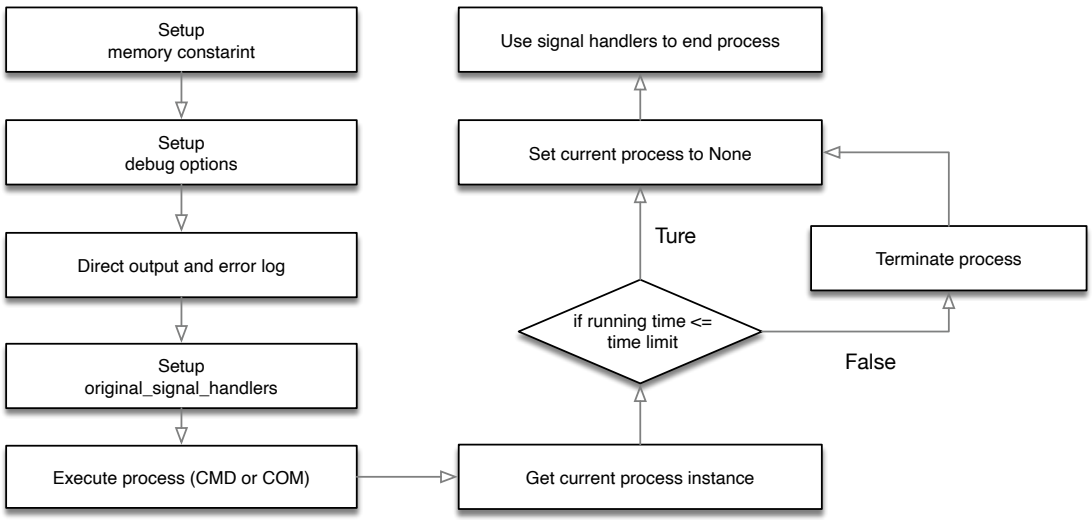


Figure 4.8 The flow logic of simulation process management

A couple of underlying reasons for the separation of simulation engine components and use RESTful style API. First, the simulation task is computation-heavy and time-consuming work; it is not likely for users to get a large network result within a short amount of time. A dedicated simulation that only receives the user's request processes the know-how simulation task then sends back the simulation results is more practical since it would not take up too many cloud computing platforms' computing resources. Second, mainstream simulation software is commercial that might need a license or dongle to be active, thus limiting the simulation engines' scalability. It is often unlikely to install those licenses and dongle in a Unix system; therefore, the most feasible method is to equip a set of process management functions that could be invoked with RESTful API to the computers that have those valid simulation engines installed. Connecting those computers to the network, configure and turn them into simulation servers. In this context, the labs that installed valid simulation engines would significantly improve the usage of their commercial license and computing power to provide simulation service to the potential user on the cloud computing platform. Finally, the RESTful API promotes a stateless API pattern to increase the reusability and applicability of API by limiting the user status involvement within the server's response process. Therefore, the requests from the cloud platform could be a collection of various APIs to define a specific request thoroughly; the simulation servers need to provide instant feedback to those calls. The general design of the RESTful API for the simulation engines is shown in Figure 4.9, where the user’s request and response are processed through the API endpoint and structured in JSON format.

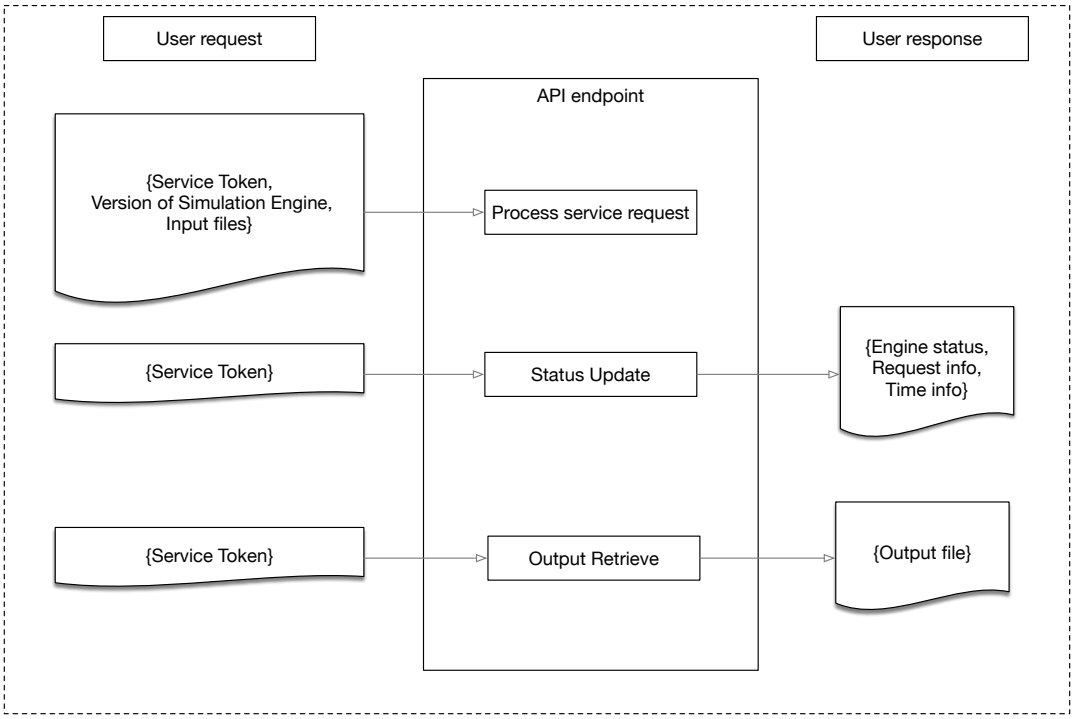


Figure 4.9 The API endpoints of a simulation server

Three primary endpoints are designed in a WSGI (Web Server Gateway Interface) server to receive client-side requests. Starting with the user request, a complete service call could be classified into three stages as the middle three API endpoints shows: initializing a service process, getting the service status, and retrieving the output. For example, when a user tries to initialize a new service request, a request JSON input would be directed to the Process Service Request endpoint. If no error occurs, the simulation server will start a new process through the predefined CMD/COM interface. The WSGI server maintains the ongoing process and server status in the backend, which can catch any exceptions and errors and ensure output generation. It is a worthy note that the Process Service Request endpoint would not respond until the process is completed.

After completing a simulation run, it would be redirected to the Status Update and Output Retrieve endpoints for further interaction.

Although there is no direct response from the Process Service Request endpoint, the client-side can check the process status by access the Status Update endpoints. Under this scene, the user could get two classes of information in a JSON format. First is the Engine status class, which includes the server status code and the run time information. The server code is built on top of the HTTP response status codes and combined with few custom codes. The expected status codes cover 200 OK, 400 Bad Request, 429 Too Many Requests,501 Not Implemented etc. On the order hand, the run time information would list the start time, running time, and final time for a simulation run and the engine idling time when no task is under running. Second, request information contains the basics description of the simulation input such as project name, file size, engine version, service token and so forth.

Depending on the simulation request information, the client-side would get an estimated running time for the service. It could use the estimated time to set up an access frequency to check the access to the Status Update endpoints iteratively. Whenever the user gets a 200 OK status instead, the client-side may send a GET request to download the simulation engine server's simulation output. Besides, with the variety of the simulation engine configuration, the computation capability and engine availability differ from server to server. Therefore, the cloud middleware is developed to match the client requests to the proper simulation server.

4.3 Cloud Middleware

The cloud middleware is an intermediary component between the service tools and the available simulation servers. The mission of the proposed cloud middleware is to match, build, maintain a two-way communication tunnel for the service client and server. Accordingly, the designed components could also be illustrated in three layers. Figure 4.10 depicts the interrelations between the user requests, simulation server, and functions in cloud middleware.

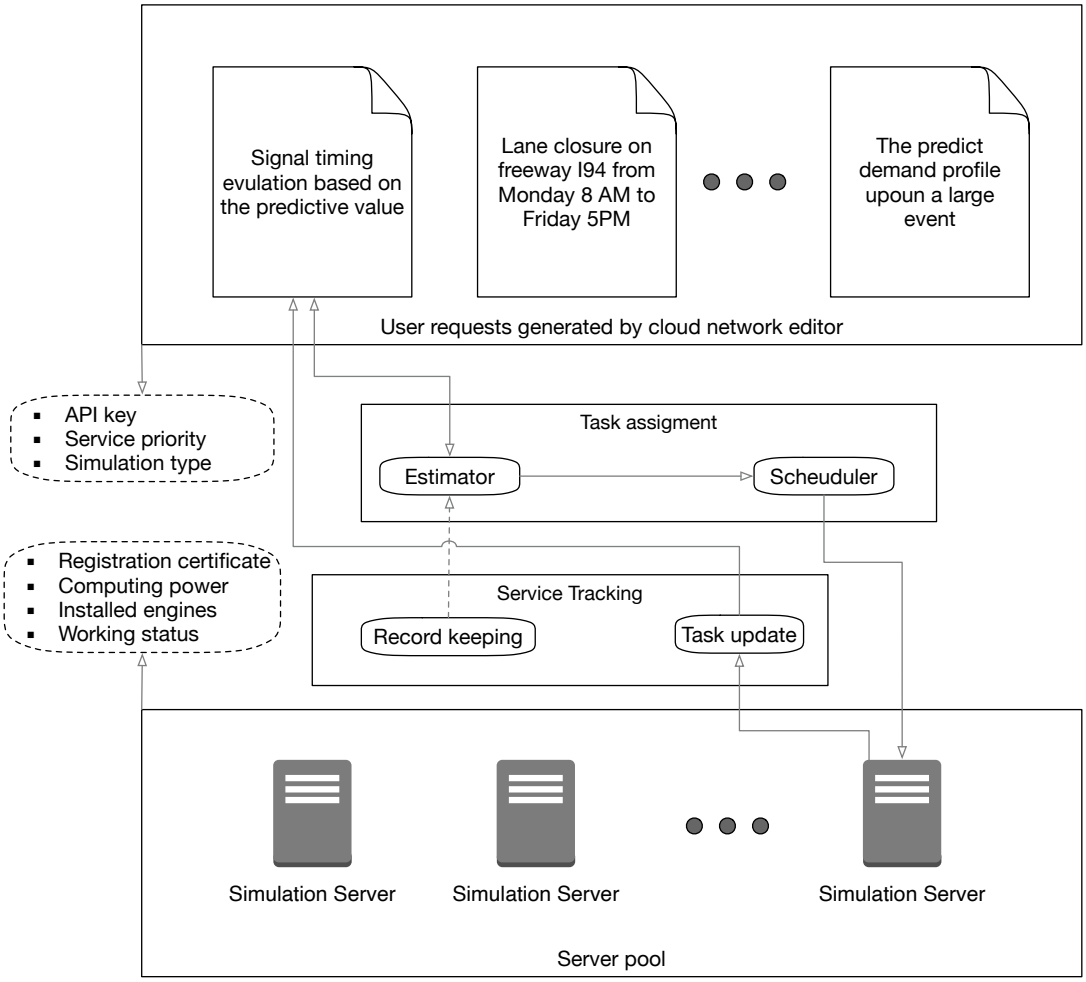


Figure 4.10 The functional design of the cloud middleware

The bedrock of a cloud middleware is a server pool that collects a group of qualified simulation servers. The simulation servers equipped with the API, as mentioned earlier, architecture would have the qualification to be registered with a certificate issued by the cloud middleware. The certificate specifies the basic information such as the server's computing power, the list of installed simulation engines for the further demand-supply matching purpose. Besides, the server pool can monitor the operational status of all the registered simulation servers through the RESTful API. In brief, the role of a server pool is to maintain and report the status of simulation servers. User requests sent through the cloud-based network editor would be encrypted with a dedicated API key, facilitating user identification and authorization from the demand side. In addition, depending on the user-selected user service plan, the level of service priority would be labeled to its request along with its simulation type. Combined with the original simulation scenario, that information is crucial for the cloud middleware as the input from the demand side.

Aside from the demand management and supply management module, the most critical part of the cloud middleware is the load balancing module consisting of two vital submodules (task assignment and service tracking, depicted in the middle two blocks of Figure 4.10). Triggered by the service request in the request queue, the estimator function in the task assignment submodule would take the user request inputs, evaluate the previous service records, and provided the user with the estimated range of simulation time. If the user confirms a service order, the plan would be transferred to the scheduler to handle the queued requests and available simulation resources. Once the client-server relationship is built, the task update function in the service tracking submodule would also be available to the user to get instant feedback on the status of the simulation

task. An assignment algorithm has been developed to minimize the service waiting time to serve the user better in this context.

Task scheduler

The availability of the simulation engines is tied to the operating system and subscriber resources. Thus, the resource pool consists of a number of loosely coupled computers installed with Windows OS and proprietary simulation software, which suits a distributed system rather than a grid computing system. In contrast, a grid computing system is the most frequently adopted architecture for business in the Cloud sector since it aims to utilize ideal CPU cycles and storage of millions of computer systems based on homogeneous resources. However, the heterogeneity among simulation tasks also poses a challenge in determining the task processing time over particular ad hoc simulation engine specifications. Frequently, a long-lasting task would lead to starvation or an indefinite blocking problem. Therefore, before a large simulation is run, a minimum viable task scheduling model, designed for non-preemptive simulation tasks is needed, which is described as below:

Objective Function

The objective of the distributed and non-preemptive task scheduling problem is to minimize the overall waiting time of the queuing simulation requests; the function can be given by:

$$Min: \quad \sum_{i=1}^{P} \sum_{j=1}^{E} \sum_{k=1}^{S} x_{i,j,k} \left(\tau_{i,j,k} - t_i \right) \tag{1}$$

All the of definitions and notations used in the Mixed Integer Programming model are summarized in Table 4.1.

Table 4.1 Notations used in the MIP model

Indices	
i	Simulation service request indexed by task
j	Subscribed simulation software index
k	Distributed simulation engine index
Sets	
P	Set of tasks
E	Set of engines
S	Set of servers
Parameters	
P_c	The load capacity of a server during a certain amount of time
t_i	Arrival time of task i
A_{jk}	Binary variable when value is one represent the engine j is installed in server k
T_{ijk}	Estimated processing time (derived from factors such as task, engine, and server)
De_i	Target engine type in task i
Variable	
$D_{i,j,k,}$	Task processing time in engine j by using server k
$\tau_{i,j,k}$	Starting time of engine j based task i in server k

Decision Variable	
$x_{i,j,k}$	$= \begin{cases} 1 & \text{if task } i \text{ is assigned to engine } j \text{ in server } k \\ 0 & \text{otherwise} \end{cases}$

Constraints

The general form of constraints for the model, except for the inner connections among parameters, are illustrated below.

Each user request for a specific simulation engine should be captured by a single available server, given by:

$$\sum_{j=1}^{E} \sum_{k=1}^{S} x_{i,j,k} = 1, \quad \forall i, i \in [1, P] \tag{2}$$

The number of tasks assigned to the engine-specific server is based on the load capacity of the server, given by:

$$\sum_{i=1}^{P} x_{i,j,k} \leq P_c, \quad \forall j, j \in [1, E] \quad \forall k, k \in [1, S] \tag{3}$$

From the demand side, the scheduler needs to ensure the task is assigned to a specified engine, given by:

$$\sum_{k=1}^{S} x_{i,De[i],k} = 1, \quad \forall i, i \in [1, P] \tag{4}$$

From the supply side, the scheduler requires the assigned server is set up with an engine capable of completing the requested task, given by:

$$\sum_{j=1}^{E} \sum_{k=1}^{S} x_{i,j,k} \cdot A_{jk} = 1, \quad \forall i, i \in [1, P] \tag{5}$$

Furthermore, if a task is assigned, the computing time is estimated by the available engine and server, given by:

$$\mathrm{D_{i,j,k}} = T_{i,j,k} * x_{i,j,k}, \ \forall i, i \in [1, P] \quad \forall j, j \in [1, E] \quad \forall k, k \in [1, S] \tag{6}$$

The starting time constraint ensures the server can only begin a new task once the previous task is complete, given by:

$$\tau_{i+1,j,k} - \tau_{i,j,k} \geq D_{i,j,k} \quad \forall i, i \in [1, P-1] \quad \forall j, j \in [1, E] \quad \forall k, k \in [1, S] \tag{7}$$

Last, the starting time of a task should be no earlier than its arrival time, given by:

$$\tau_{i,j,k} \geq t_i \quad \forall i, i \in [1, P] \quad \forall j, j \in [1, E] \quad \forall k, k \in [1, S] \tag{8}$$

4.4 Error Handling Mechanism

Dealing with complex logic and heavy input software, the design of a simulator requires simplifying the input scenario and the flexibility of combinations. In addition, an intuitive user interface design would help users reduce the effort for data entry, and the autonomous batch handling process would avoid some repetitive work. However, errors still could arise when a user makes a mistake in the input system, especially for a component-based online simulator. Thus, an error handling mechanism is implemented to guide the user to the correct format and result.

Three types of built-in error handling methods have been designed. The first layer of error reporting embedded in the input data containers include file instance and data model. The interface for file instance would check its file type and file size at this stage. In some cases, the file instance, like some excel templates, has built-in functions to validate logical inputs. However, in most cases, the logical check and content check would be processed in the second layer of error reporting, named the program instance validator. The data model error checker would collect the input formation structurally and check the predefined data type in each input box.

The program instance validator is a module of scripts that handle the data input from the data model and file instance, further process, extract, recompile those data components into a

specified file format, and then send them to the simulation engine. During this process, errors and bugs often arise since the open-ended data input source is different from the traditional integrated software distribution. Traditional software has a standardized user interface to limit the user input flexibility and minimize errors. The open-ended data editor would exponentially increase the number of potential errors. However, when the data editor is combined with a program instance, the errors and bugs could be well-captured by the service provider and use as inputs to advance the robustness of its program instance. At the second stage, the cloud computing platform would create a session-based terminal for the service provider to receive all the feedback caught by the python interpreter. In addition, the service provider could leverage the use cases to improve the error handling functions in the program instance. The red rectangle boxed area in Figure 4.11 shows the user interface in the view of service provider of the session-based terminal.

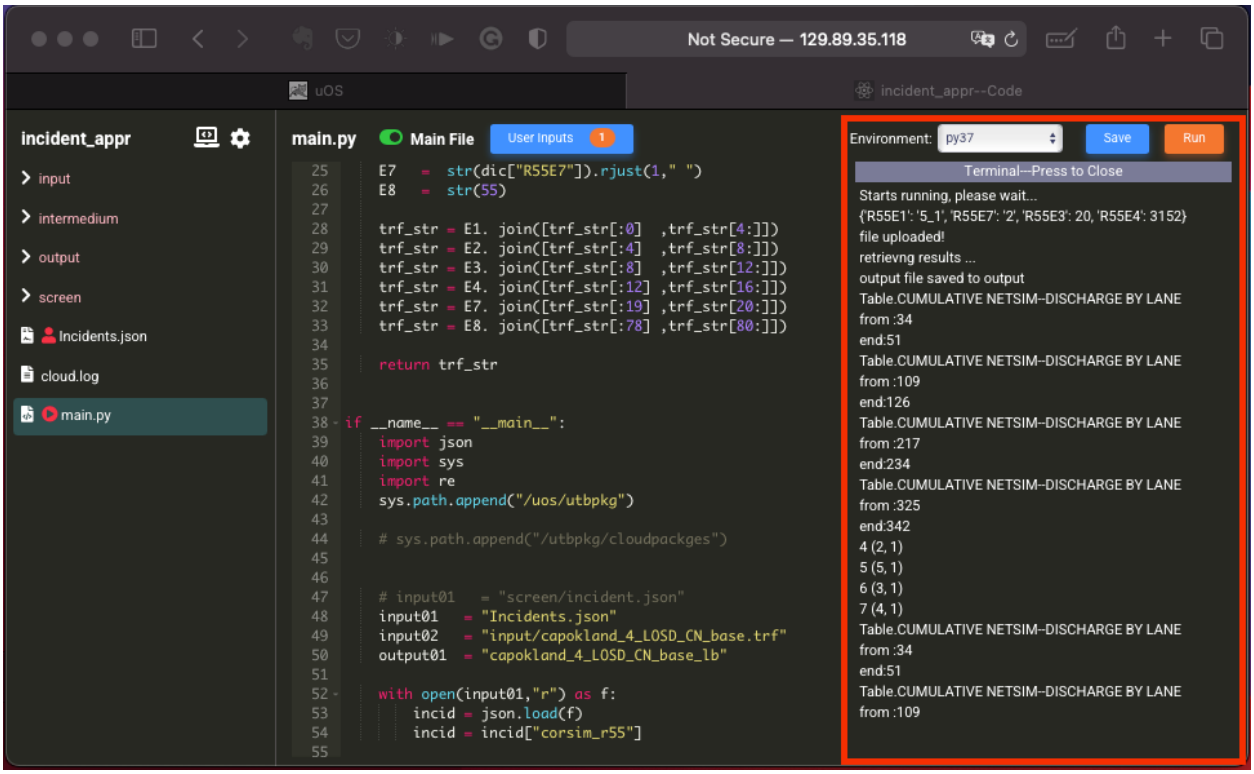


Figure 4.11 Session-based terminal to catch the program error

After the program instance completes the compilation, the third layer aims to capture errors that would occur with the simulation engines by processing the standardized input files. The errors at this stage could be classified into warning messages and fatal errors (*Warning Messages and Fatal Error CORSIM*, n.d.). The warning message stands for unusual conditions that may indicate an error. The occurrence of a warning message will not abort the run. Usually, this message could be prevented in the previous two error handling layers. A fatal error represents conditions that would abort the run, which means the simulation will not be performed. At this stage, the errors would be stored in the simulation server to be accessed through the designed RESTful API and presented to the user.

In this design, even though a user has successfully passed the first two stages in the simulation life cycle, the error message at the third stage is typically too complicated for a user to figure out and debug. Sometimes, experienced engineers and technicians need to try multiple solutions to debug the problem. In practice, entry-level technicians could seek help from some senior engineers, and they would form a small community in the consultant service area. If no viable solution is found, they would turn to the software provider for technical support. However, these practices have not been documented and shared within the larger communities. People are plagued by similar errors that consume time to correct. Therefore, a large-scale social involvement is needed to collect more professional experience and intelligence on the cloud platform, which is demonstrated in Chapter 5.

4.5 Conclusion

This chapter introduces four critical components (cloud-based network editor, simulation engine, cloud middleware, and error handling mechanism) of a customized online simulator. This section examines the qualities of the software architecture concerning the following criteria.

- **Performance**: The existing integrated software uses local memory to store essential information and streamline the entire lifecycle, which provides an advantage in the network communication assessment since it needs less throughput than the cloud component-based simulator. However, the user-perceived performance is primarily dependent on the user's scenario and the component architecture is considerably more flexible for entry and medium level scenarios with no noticeable loss in performance.
- **Scalability**: integrated software is scaled as a product that sometimes needs a license or physical

USB dongle. The component-based architecture allows the cloud-based network editor to scale by purchasing a program instance simulation engine and scale through a server's configuration. In addition, the program instances that interpret and visualize the simulation outputs can also be scaled, as will be presented in Chapter 5.

- **Simplicity**: integrated software often comes with a stack of documentation which represents a steep learning curve. However, the learning process of a technician cannot be replicated. If a user needs to use five percent of a software package's function, there still exists a steep learning curve. On the other hand, the component architecture allows the service provider to tailor its components to a lightweight cloud application, which minimizes the users' learning effort.
- **Modifiability**: the ease with which a change can be made to a simulator architecture. It can further be stratified into evolvability, extensibility, customizability, configurability, and reusability. Since the proposed cloud simulator is component-based, it has significant advantages over integrated software in all five measures.
- **Visibility**: the interactions and running status between components is hard to monitor in traditional systems. Some integrated software allows technicians to use the COM interface to capture the running cache, but this level of monitoring is highly unlikely for a typical user. The cloud simulator provides an error handling mechanism and cloud middleware to easily present users with instant feedback.
- **Portability**: Most of the integrated software only runs on Windows platforms, except SUMO running on UNIX, macOS, and Windows. The cloud simulator grants users access to the UI in every device if the device has a browser and internet access.
- **Reliability**: the integrated software should be more reliable due to its long-term distribution and

professional technical support. However, the cloud simulator is a service-oriented software; the reputation of a program instance reflects the service's reliability.

Chapter 5 Platform Modeling

This chapter illustrates the platform interaction model and value proposition of the cloud platform for facilitating the transition from component functions to simulation services. According to the basic service logic described in Chapter 3, the value proposition that the platform brings to the user and service provider could transpire along three stages: Independent processing, Surrogate interaction, and Direct interaction. Beyond a specific simulation service, the output management and community building mechanisms are discussed with the goal of reducing data silos.

5.1 Service Development and Operation

Like any physical or virtual service, the service provider and user activities can be classified into three categories: Independent processing, Surrogate interaction, and Direct interaction. The platform handles these critical activities. Thus, both the service provider and the user would choose the platform as the trading place. This section breaks down the critical activities into the three categories to demonstrate the advantages of the cloud platform.

5.1.1 Independent processing

The independent processing happens in different places for service providers and simulation users. In terms of a demand-oriented service, a potential case would be the government post an initiative for a new activity, process, or form. Then, qualified consultant companies would bid for the contract. The service provider and user will approach this new initiative independently until they converge and see the benefit of collaboration. With a more fine-grained look, when the consultant companies need to apply their solution to a real problem, they will refer to their organizational experience and existing functions for assistance. In this case, the service is provided

by the domain knowledge provider, either the manual, scripts, or software. Its paradigm has shifted to a supply-oriented service. The approach of the people developing the service is to find a feasible solution by leveraging their tools and experience.

The cloud platform, developed primarily by service providers at a granular level, follows a bottom-up approach in knowledge acquisition; thus, it can be positioned as a supply-oriented or developer-generated platform in a general sense.

The service provider or application developer is the central participant in the creation of the cloud platform. They need to develop, test, and deploy every tool used in private or public. If the tool is a program script for self-use, the code will likely not be a professional code, but simply a method of solving the problem. However, those scripts are hard to maintain on a local desktop, especially when the number of scripts increases due to iterations. Also, new technology becomes available every day, and old scripts will get outdated if they do not keep pace with evolving dependencies. Therefore, encapsulating the scripts, and deploying them for public evaluation, is essential to take advantage of the built-in value and promote the updating of core functionalities for all users.

Despite that a portion of service providers recognize the importance of publishing their knowledge and functions, the techniques in full-stack software development are non-trivial and experience-intensive. There lacks a convenient way for those providers to serve their developed functions on remote sites instead of adopting a code-on-demand mode, which requires the end-user to have an advanced knowledge on how to run the code locally. Although some excellent developers can complete the entire full-stack development process, the considerable cost of hosting the service would require a feasible monetization scheme to enhance its sustainability.

The cloud platform provides two modules: Cloud Storage and Social Hub, to address previously mentioned practical issues. In the independent processing stage for the service provider, the Cloud Storage module would ease the process through critical activities such as virtual environment management, version management, repository sharing and collaboration, session-based remote evaluation, and public URL assignment. Note that the cost-of-service hosting in the session-based remote evaluation approach mentioned in Chapter 4 is significantly lower than the virtual machine hosting solution. The Social Hub, on the other end, would bring traffic to the published post and set the monetization scheme for the service provider. In this sense, the value propositions that the cloud platform brings to the service provider may involve improved knowledge accumulation and management, improved collaborative development, remote accessibility, service reliability, and publicity.

From the user's viewpoint, the activities primarily occur on the Social Hub module since neither the demand posting nor service searching is a socialized activity. In the Social Hub section, the user could post their questions or requests, and then the cloud platform would leverage its underlying searching algorithms to feed the demand to potential qualified service providers. When it comes to a user who wants to find a suitable tool to solve a specific issue, the Social Hub can also provide potential service tools based on their evaluated ranking. The Social Hub is ideal for domain communication and toolset discovery since it supports subscription/follower relationships and content recommendation mechanisms. The archived content on the Social Hub can be accessed in the Cloud Storage module's private space.

Derived from the key activities, the value propositions from the user's perspective could be summarized with the following points: qualified information feeds, evaluated service feeds, domain practitioners' community, and improved service discovery process.

5.1.2 Surrogate interaction

Unlike independent processes, which act only on resources owned or controlled by the process entity, surrogate interactions include activities that are not owned or controlled by the entity, without direct interaction with the owner (Sampson, 2012). To be more specific, the surrogate interaction is a response process for the service provider triggered by a detailed user's request. With such resources (user inputs and requests), the service provider can start designing and assembling the workflow, tuning the available functions accordingly, and preparing a contract, without customer involvement at this point. Therefore, the surrogate interaction steps often occur in the project development process for both service providers and users before they head on to the project negotiation and direct interaction phase of a project.

Service providers may use local resources and applications on their desktop to produce the deliverables. But they will choose to use Cloud Storage on the platform because it is a collaborative place to deliver outputs to the user side. They can leverage the repository sharing and collaboration mechanisms to simultaneously manage various inputs and outputs, which is especially beneficial for medium and large simulation projects. On top of that, the service provider could seamlessly config the inputs and direct the outputs of its proprietorial server tools on the targeted cloud repository. Thus, the purchased service tools could be utilized to serve the project on demand and bring in cash flow. The cloud platform could take a portion of the project funds to feedback to the service tools developers, attract more developers and enhance the service life cycle.

Moreover, the primary advantage of Cloud Storage is that it allows the service provider to assemble the service tools either purchased or self-developed flexibly, form a customized workflow, which is reproducible, shareable, and purchasable. The design of a simple workflow is illustrated in Figure 5.1, is a reusable graphical illustration showing a set of distinct service tasks and specifying their interconnection. The workflows connect the task objects and form a directed acyclic graph (Hart, 2011), which should be specified by the service provider and guide the user to execute the output correctly. In addition, each task object is a self-contained program that was used to perform a specific step towards the final service. The critical difference is that this workflow focuses not only on the execution of tasks but also on guiding users, ensuring visibility, and service engagement.

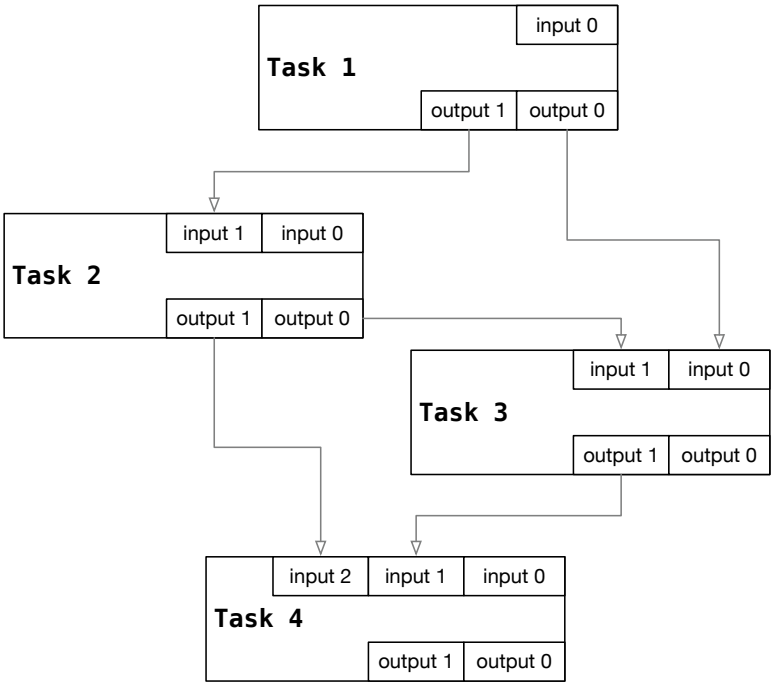


Figure 5.1 A simple graph illustration of a customized service workflow

Upon completing the designed tasks in a workflow, the service provider will benefit from the cloud platform by streamlining the data service experience, improved project delivery, increased number of service tools, enhanced automation levels, increased flexibility in cost and process control, and the provider will have a reproducible workflow. On the other hand, the user will enjoy value propositions mainly regarding the high degree of freedom in customization, pay as you need support, instant project feedback, ease of sharing, and quality of presentation.

5.1.3 Direct interaction

Communication skills consistently rank high in the skill list from engineers' views or managers' priorities (Kinney & Ra, 1995). As an essential factor of direct interaction, communication conveys the level of customization and service. The direct interaction steps involve a process entity working in conjunction with one or more other process entities. To account for the efficiency of the direct interaction process, the cloud platform develops two types of operation desktops built on top of the previously mentioned sharing and collaboration mechanisms. Therefore, from managers to entry-level engineers in a cross-department project, participants could work simultaneously with various views, but clear task assignments, to reduce the peer to peer, top to bottom, and bottom to top communication costs.

One of the operation desktops is the Code-Oriented workspace. The workspace refers to the combination mechanism of the cloud computing virtual environment and the cloud storage. Driven by the cloud platform's mechanism, the cloud storage is designed to support the tree-based file system and instantiate the run time environment based on the working directory. Each account will have its own cloud storage space and a limited number of virtual environments; thus, the user or the service provider can run a session-based process under the pre-set environment. The desired

deliverables could then be generated autonomously and accessible to the stakeholders, which completes the service cycle of request, execution, and generation. Remarkably, all the service operations may be completed offline or through an existing cloud computing solution, but the interaction between the user and service provider is not in a direct and rapid approach. Distinct from the existing methods, the developed workspace's primary goal is to facilitate direct interactions such as file operations, code executions, and project management. Therefore, the developed cloud platform has a sharing and permissions mechanism for the workspace to define each participant's privileges. For example, in a simple simulation service, the user may have permission to run code on the cloud that follows the same user experience as the provider. However, the service provider has permission to hide the code and only give the code to users with executive privilege. From the user's view, the execution button acts like a customized function in a black box. In this sense, the service provider could keep the code confidential, and the user can get a higher degree of customization and granularity on the cloud platform.

Map-Oriented workspace is the other option built on top of the sharing and permissions mechanism and tree-based file system. Code-Oriented workspace focuses mainly on the execution side, whereas the Map-Oriented workspace pays more attention to manage files and interfaces. To some extent, it equals a graphical user interface built on top of the map objects. Figure 5.2 illustrates the file structure and functions of that Map-Oriented workspace.

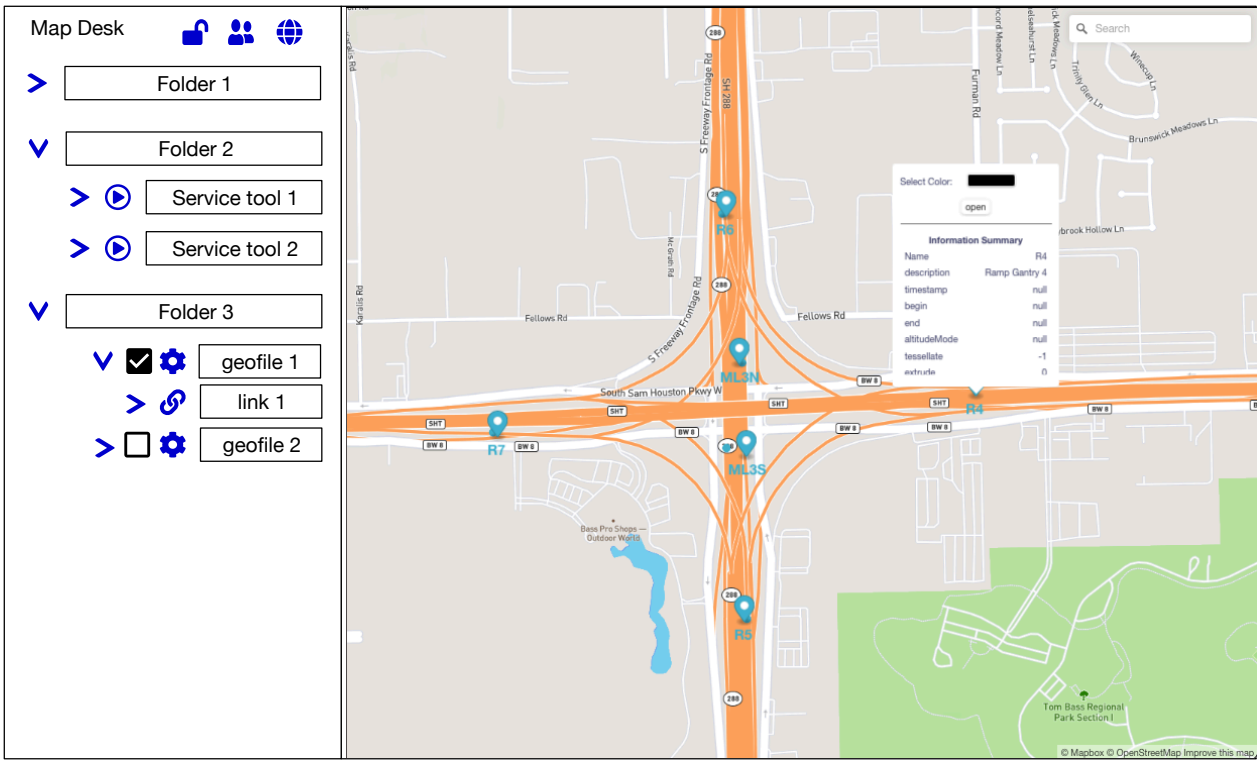


Figure 5.2 The conceptual model of the Map-Oriented workspace

As shown in Figure 5.2, the user can see a tree-structure management panel on the left and a map navigator on the right. Any map-oriented workspace in the cloud storage can be opened and viewed in the map mode. Critical features like locking mechanisms, member permission control, and styling can be embedded in the folder. The service provider can control files by specific names. The unique components supported by the Map-Oriented workspace are introduced and described below:

- **Geofile**: a georeferenced file that can be converted to a set of GEOJSON objects. If the box in the left panel is checked, like geofile 1 in Folder 3 shown in Figure 5.2, the map section to the right will render the graphical object in the map and support objects level operations such as

data collection, program execution, and result management.

- **Acquired program instance**: a program instance developed by a service provider and acquired by a user. When a program instance is published and listed in the market, it can be purchased by users.
- **Symbolic link**: a link object can map the GEOJSON object to any other objects stored in the users' cloud storage. It enables the service provider to integrate a resource efficiently and intuitively while providing the user with a straightforward and compact graphic interface for browsing and management.

In conclusion, workspaces are for direct interaction and enable both user and simulation service providers to submit, execute, and present ideas quickly and responsively. The workspace provides a testing ground for developers to iterate their code and improve the code's robustness. In terms of the user side, the user will enjoy customized services with a higher level of automation; thus, the service cost will gradually decrease in the long run.

5.2 Output Management and Representation

According to a definition by Fielding, a representation is a sequence of bytes, plus metadata to describe those bytes (Fielding, 2000). The representation of simulation outputs includes plain text, documents, files, figures, reports, and animations. This section introduces an interactive dashboard that builds upon those bytes in output files to help end-users understand the critical metrics regarding the simulation results.

The benefits and needs of establishing a dashboard for both long-term process and short-term performance measures at the state and agency level have been acknowledged and

accomplished (Hranac & Petty, 2007). The maturity of the underlying database technologies and web technologies inspired the design of the dashboard. The long-term process dashboard has been broadly accepted to monitor the project budget and schedule metrics architected on the data warehouse and client-server solution. The short-term dashboard is gaining broader acceptability in operational performance measures as the cost of data warehouse and client-server solutions decline. The Regional Transportation Commission of Southern Nevada (implemented a web-based real-time freeway and arterial performance measurement system developed with Microsoft's Visual Web Developer 2010, Microsoft .Net technology, Silverlight, and Bing Maps (Xie & Hoeft, 2012). Inherited by the client-server architecture, a dashboard tool was developed to communicate the cost related to delay at the international border crossings (Rajbhandari et al., 2012). The California Department of Transportation (Caltrans) embedded two tools which are the continuous risk profile (CRP) and the California Safety Analyst (CASA), in its web-based dashboard system to enhance functions for safety management (Chung et al., 2013). Although the existing client-server architecture works smoothly in various states and agencies with sufficient budget and technical support, a concise dashboard's deployment cost remains too expensive to scale, limiting its accessibility for small agency or project-level representation.

5.2.1 Scalable representation architecture

Regarding project-level service such as a microscopic simulation, the output file size is very large and too complicated to analyze without the help of tools. Thus, a postprocessing algorithm is needed to bring an integrated view into the dashboard for better understanding by the end-users. However, a project's scope and the nature of a simulation are viewed very differently by the agency-level management team making financial decisions for an organization. The project

scope prevents the implementation of a client-server architecture due to its one-time delivery style and the large cost. Also, simulation is an iterative process that requires multiple runs and produces multiple versions of outputs for the stakeholders to evaluate. Therefore, the centralized data warehouse is not suitable for a constantly changing data source and iterative visualization style. In terms of visual analytics at the project-level data granularity, this study proposes three types of dashboard architecture that are compatible with the cloud computing platform to reduce the deployment cost and increase the scalability of the dashboard.

The first architecture is a document-based solution that allows the dashboard to be manipulated and transferred through a single HTML file. In this case, the HTML file could be shared as a document, sent to multiple stakeholders, open locally for simulation result evaluation. Meanwhile, the HTML file is shared in the cloud computing platform as a URL link, and multiple stakeholders can simultaneously access the web-based dashboard. The file structure and visualization mechanism of a document-based solution as illustrated in **Figure 5.3**. This dashboard container is an HTML file consisting of Document Object Model (DOM), Cascading Style Sheets (CSS), and JavaScript. The configuration setting and input setting in the metadata module are designed to generate the HTML skeleton template. In this case, the metadata about the configuration can determine the layout and interaction of the dashboard components through the manipulation of DOM and CSS. In addition, the input metadata contains the chart appearance options and the series data that is injected into the JavaScript section. Therefore, the HTML file is filled with a structural representation of style and raw data that creates the dashboard. Then, when the HTML is opened under an internet-connected environment, the JavaScript section requests

server-side visualization render the chart component and send it to the corresponding DOM location.

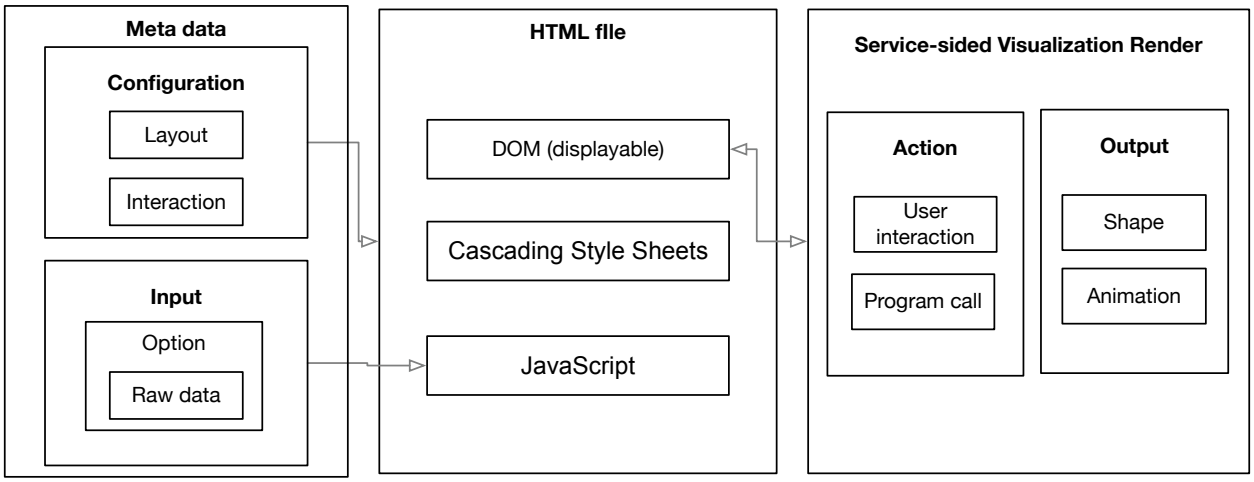


Figure 5.3 Single document dashboard architecture

Note that the server-side visualizations involved in this solution could be Echarts and KeplerGL. ECharts employs an all-in-one hierarchical JSON format option to declare the components, styles, data, and interactions, resulting in a logicless and stateless mode that enables easy sharing between collaborators and stakeholders (Li et al., 2018). Moreover, KeplerGL provides a client-end geospatial analysis framework to explore and filter the CSV format and GEOJSON format data in the browser (*KeplerGL User Guides*, 2021).

However, one of the document-based solution's main disadvantages is that the user interactions with the displayable objects in the chart are limited within each DOM object. In other words, the displayable object in DOM object A cannot interact with DOM object B. Therefore, the single document representation is suitable for small-sized dashboards such as intersection level analysis but remains hard to represent medium and large-sized network data.

To address the issues of the document-based solution, a folder-based solution is developed for medium and larger-sized networks by leveraging cloud storage. This second architecture takes the folder as the representative unit and uses a tree structure for organization. The folder architecture stores the supportive file instances, and declarations, in the metadata in an HTML index file. The architecture of the folder-based solution is illustrated in Figure 5.4. Unlike the document-based solution, the rendering of shape, animation, and the execution of the event handler does not rely on the third-party server-side visualization, but instead uses the cloud computing platform. In other words, the cloud computing platform could use its cloud storage section to host a web service that is organized in a folder manner. Users and collaborators can share, duplicate, and scale the folder-based solution at ease in this cloud storage context instead of undergoing the current complicated code-on-demand process. For example, a code-on-demand platform like GitHub requires users to download the code to the client-side, set up its environment and dependency, and run a server locally.

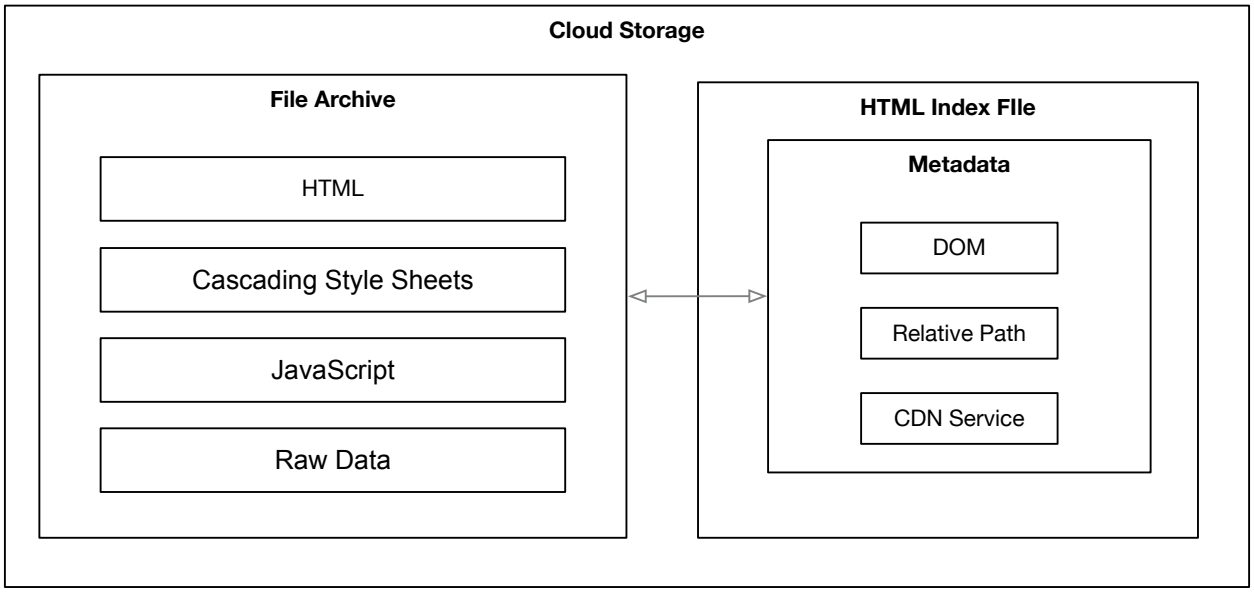


Figure 5.4 Folder-based dashboard architecture

Looking closely at the dashboard data flow mechanism, the cloud hosting feature enables the developers to employ the JavaScript API to load raw data into the client-side and store it in the browser's memory.

In order to design an informative dashboard, the developers need to be proficient at web development. For example, libraries like D3.js and ProtoVis could produce good quality 2D drawings, while jQuery and AJAX can provide the DOM manipulation methods through JavaScript that enhance the flexibility for user interaction. The technical stack to design, develop and debug is sophisticated and requires knowledge in HTML, CSS, SVG, and DOM, which makes the development non-trivial. Therefore, the final product should be in a ready-to-use status that is scalable on cloud storage to get more exposure.

Lastly, if a developer needs to customize the data operation functions to enhance the dashboard, a session-based solution is recommended to implement this capability without losing scalability. As it could be seen from Figure 5.5, the session-based solution is built on top of the folder-based mechanism and a session-based front-end and back-end software architecture. Under this setting, the user interaction within the dashboard is not limited to the available raw data but the interaction can be sent to a session-based backend to execute operation scripts and modify the raw data. In other words, this solution provides the dashboard with CRUD capability allowing for raw data modification that improves data processing efficiency and optimizes cloud storage.

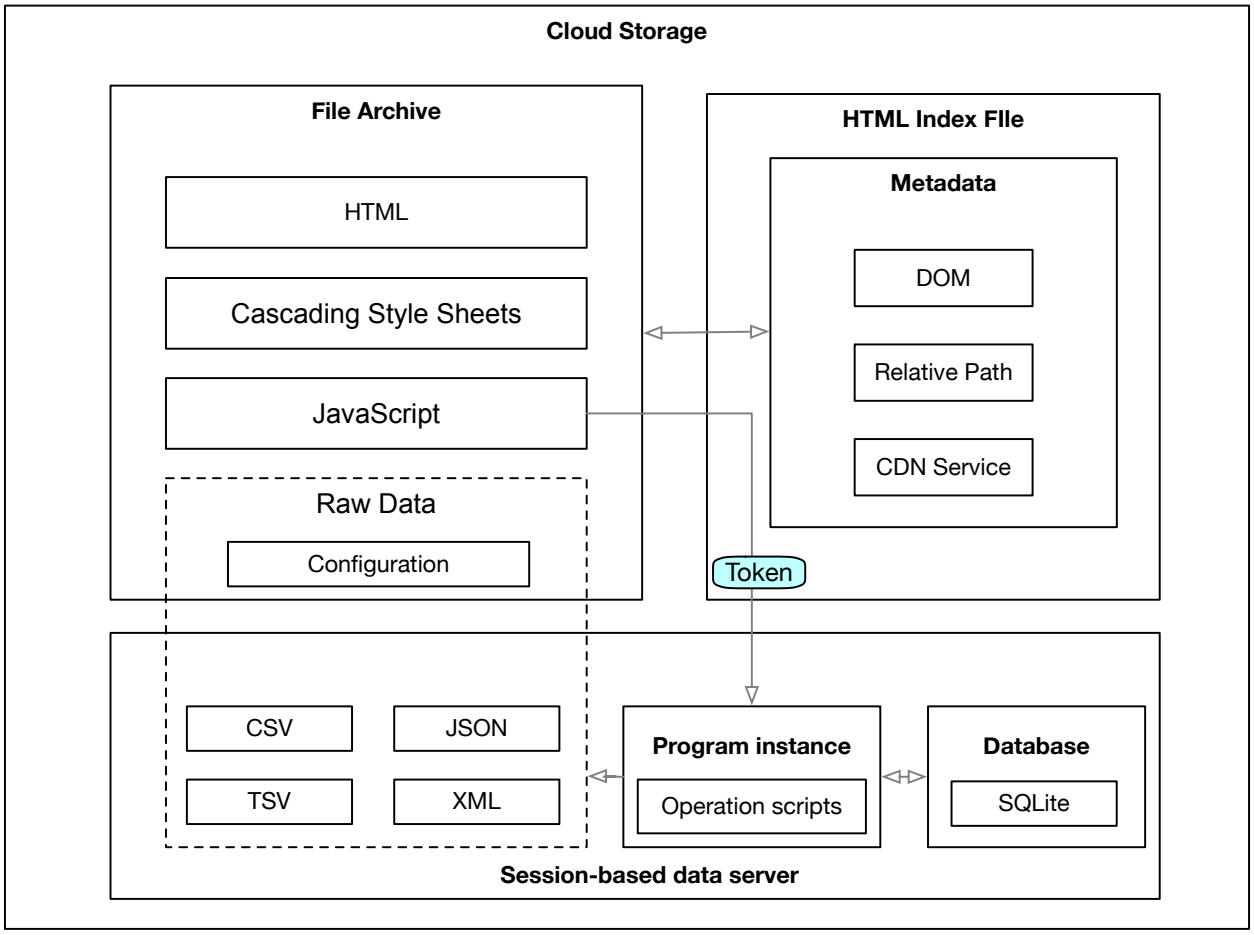


Figure 5.5 Session-based dashboard architecture

It is noteworthy that a single-file cross-platform database plays a critical role due to its self-contained, serverless, zero-configuration features. Because of the on-disk file format, it ensures better performance, portability, reliability, accessibility, as well as reduced application cost and complexity, which is ideal for the small to median sized project. Besides, since the dashboard application is a project-oriented deliverable, the data is integrated into the application. Local storage with low writer concurrency makes this structure work in an SQLite environment.

5.2.2 Dashboard design in simulation MOEs

The dashboard was envisioned to structure the performance measures using a single report (represented by charts and graphs). However, the charts and graphs' data are dynamically related in the spatial and temporal dimensions. It was proposed that the dashboard should be designed intuitively and interactively to reflect the time variation of critical indicators of the transportation facilities. The predefined granularities used in the visualization design include temporal trend, geometrical trend, and longitudinal trend. Therefore, the short-term aim of this section is to qualify and quantify what type of information is best displayed with the chart. Several case studies are devised in four aspects (design goal, user, user task, description) for the specific goals of:

- Visualize the level of service at the intersection level, which accounts for the input volume, geometry, and signal timing.
- Communicate the critical indicators of a corridor under a specific scenario.
- Examine the demand and supply of a network level and communicate measures and attributes through the map explorer.

Intersection level dashboard

At the bottom-most service, the smallest simulation unit is an intersection, which requires detailed observed or estimated input information as described before. The dashboard presents the existing geometry and signal timing information and integrates visual analytics in multiple aspects based on the simulation results.

Design Goal: Develop a report for evaluating the effectiveness of a signal plan.

User: A signal engineer in a consulting company or a college student interested in transportation engineering

Users Task: To understand vehicle movements and signal control patterns across legs in a typical intersection; study how a signal plan affects the level of service and other critical measures of effectiveness derived from the HCM method and simulation engines.

Description: An image of the single document-based dashboard designed for a typical four-leg intersection is shown in Figure 5.6. While the underlying document architecture provides its portability, the open-ended input format enhances its accessibility, and the supporting services ensures scalability. Narrowing and focusing the user on specific inputs and measures and providing direct comparisons of results are the essential benefits to the user. Therefore, the three-fundamental classifications, Demand, Geometry, and Control, are imbedded in the report structure and integrated into the blue navigation plane on the left panel. In addition, the output, also known as measures of effectiveness, comes from two sources: the HCM method and a user-specified microscopic simulation engine. The user can select a perspective by choosing the administrative widgets through a checkbox that folds and unfolds the respective sections, where the medium HCM-based column provides a static intersection image. Meanwhile, the right section (post-processing from the simulation engine) uses geometric imitations as a reference and emphasizes MOEs and temporal variations through lightweight interaction such as tooltips, dynamic annotations, and brushing. This layout design allows a user to understand the signal performance in an incremental pattern from left to right: general index panel, empirical-based quick estimation, and exploratory stochastic MOEs, respectively.

Starting at the top of Figure 5.6 and moving down the dashboard, a user first sees the top row that shows an Intersection Flow Map and two hierarchical sunburst charts. The Intersection Flow Map is intuitive in presenting the input value of channelization, proposition, and the volume of each lane group. The left sunburst chart shows the lane-specific simulated counts over a number of periods, and the right sunburst chart uses the same aesthetics to represent the corresponding queue length. It should be noted that color is used to distinguish nominal categories and serves as an index to guide users to identify each approach's position visually.

The second section down the figure displays a combination of geometry and control attributes. The static abstraction of the intersection geometry serves as an index, where each lane in the approaching leg has a phase number and movement direction. Both stage-based plans and ring-n-barrier diagrams of signal timing representation are integrated into the static abstraction.

Last but not least, at the bottom of the figure, the user can get a quick understanding of the signal control performance given the flow and timing pattern. The Measure of Effectiveness section has two static geometric representations on the left and other measures of interests listed on the right. Based on the existing geometry, the Delay-Base LOS diagram uses both color and length to represent lane-specific, and overall intersection, level of service. At the same time, the right-side simulation-based diagram provides a direct color comparison in terms of the delay level. The Saturation Comparison Diagram, at the bottom of the middle column in Figure 5.6, follows similar aesthetics to display lane-specific saturation levels from the HCM method, whereas other simulation-derived MOEs can be selected from the tab bar in the effectiveness section of the dashboard. The standard MOEs fall into five categories: Network-wide average statistics, Link-Based MOE, Lane-Based MOE, Person MOE, and Environment MOE.

.

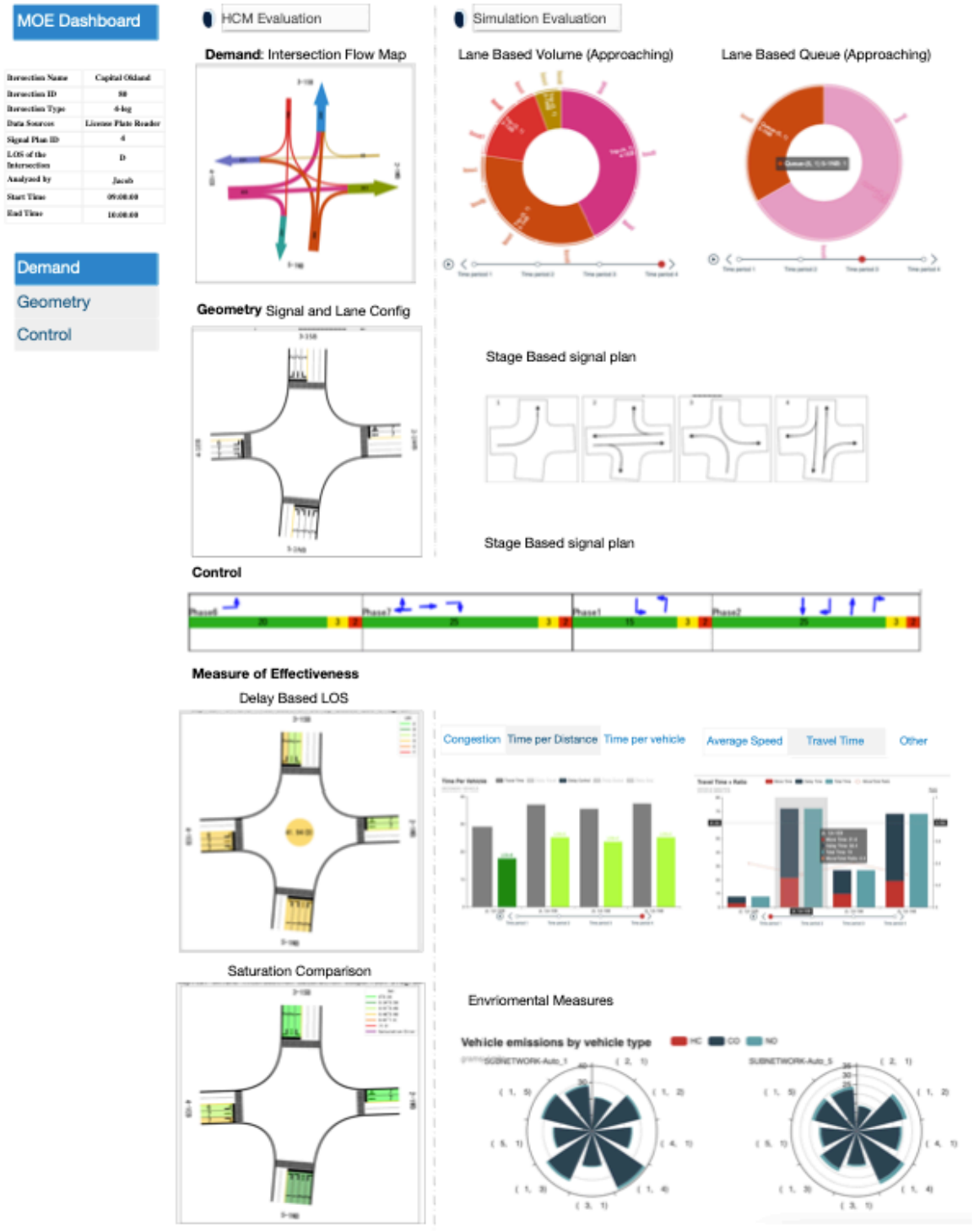

MOE Dashboard
Intersection Name
Capital Oklan
Intersection ID
Intersection Type
4-leg
Data Sources
License Plate Reader
Signal Plan ID
LOS of the Intersection
D
Analyzed by
Jacob
Start Time
09:00:00
End Time
10:00:00
Demand
Geometry
Control
HCM Evaluation
Demand: Intersection Flow Map
Geometry Signal and Lane Config
Control
Measure of Effectiveness
Delay Based LOS
Saturation Comparison
Simulation Evaluation
Lane Based Volume (Approaching)
Lane Based Queue (Approaching)
Stage Based signal plan
Stage Based signal plan
Congestion
Time per Distance
Time per vehicle
Average Speed
Travel Time
Other
Enviromental Measures
Vehicle emissions by vehicle type
HC
CO
NO

Figure 5.6 The dashboard template for an intersection

Corridor level dashboard

With sufficient intersection level input data, a microscopic corridor analysis becomes viable. However, because of the relatively large amount of data, regardless of the input preparation or the evaluation, the single document-based solution is not appropriate. Therefore, the folder-based solution is adopted to develop the Corridor level dashboard.

Design Goal: Develop an interactive dashboard to access the measures at the corridor level.

User: An engineer in a consulting company or college student interested in transportation engineering.

Users Task: To evaluate the overall performance of an operational strategy applied to a corridor, support quick visual identification of bottleneck areas, and build a connection with the specific intersection data.

Description: The layout of this dashboard is a map-centered visualization with two spaces: left-side, map and right-side, summation detail as shown in Figure 5.7. Two types of indexing mechanisms are built into the web application. The first one is a visual inspection-driven, geospatially referenced user interaction handled by the event listeners, which is designed to trigger a summary tooltip dialog that presents essential information and simultaneously updates the performance indicator charts listed in the right column. The second indexing mechanism is more effective for the experienced user and analyzer to swiftly enter the ID of the link of interest into the search box on the right top corner. During the typing of link ID, a dynamic matching algorithm is activated to feed a list of potential links instantly. When the interested link is chosen, either

mouse-click the search button or hit the enter key using a keyboard; the application will automatically locate the link, zoom in to an appropriate area, and altogether shows attributes and indicators. In addition, the right chart panel adopts a scrollable design to ensure the readability, conciseness, and variety the measures.

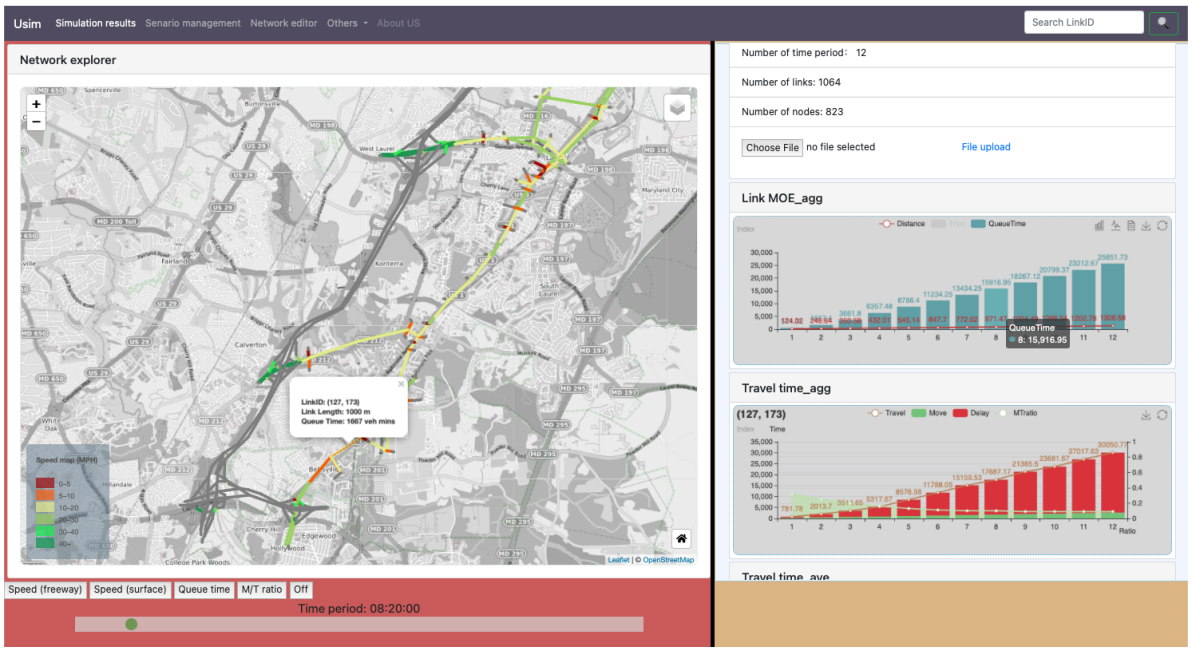


Figure 5.7 The dashboard template for a corridor

Network-level dashboard

The data preparation of a network simulation often starts by collecting high resolution data. However, the network-level analysis is primarily used in a planning level simulation, which entails a more extensive scope and only coarser resolution. Nevertheless, the number of nodes and links within a network is significantly higher than the corridor level analysis, which poses a higher bar for processing, filtering, visualizing, and evaluating. Also, the user demands for analyzing a

network vary from group to group. Thus, the session-based solution is utilized here to deliver exportability, customizability, and extensibility.

Design Goal: Provide a framework for users to assemble the functions based on the map interface.

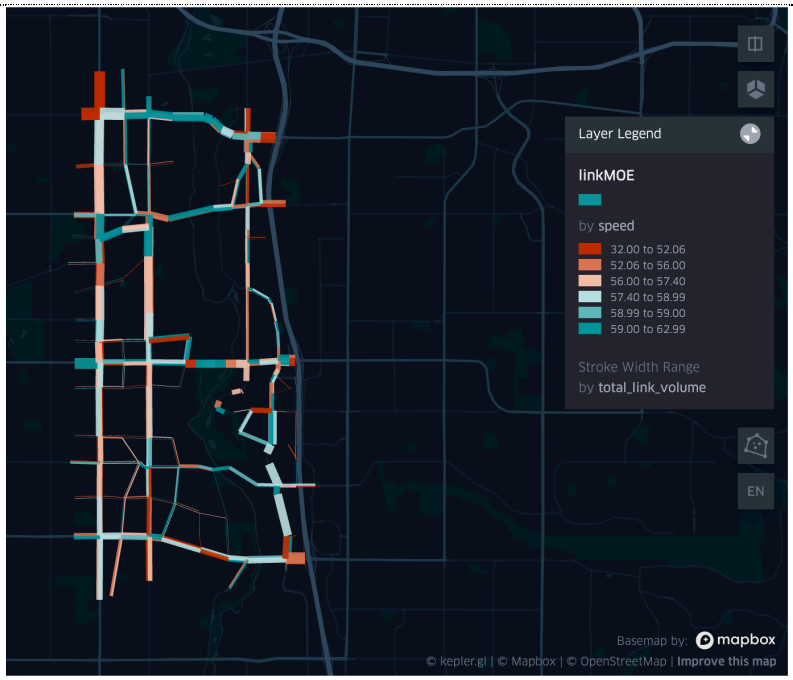
User: A transportation planner or modeler; a college student majoring in civil engineering.

Users Task: To learn the supply and demand pattern at the system level; support the data processing and exploration seamlessly; facilitate the sharing and evaluation among the stakeholders and the public.

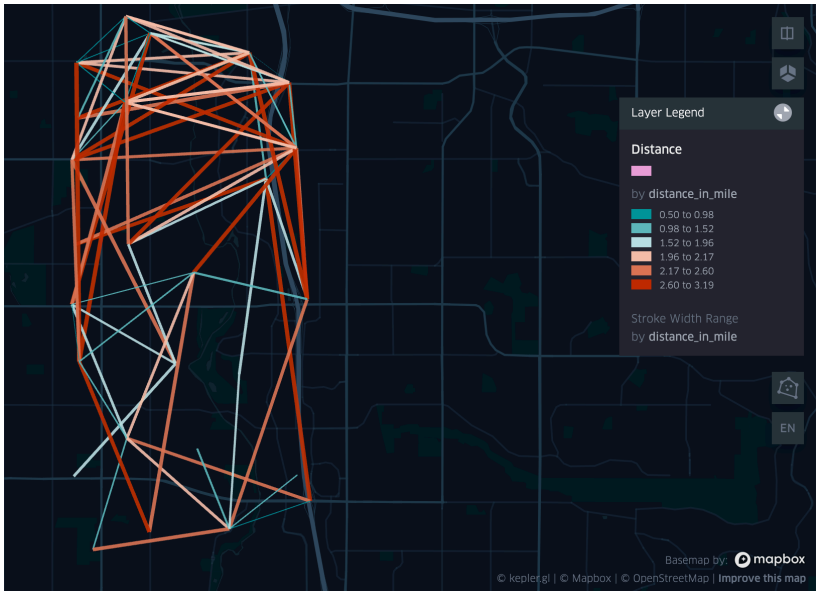
Description: Instead of using a standardized dashboard to perform the geospatial and temporal analysis, the network-level dashboard leverages the Map desktop in the service platform and program instances maintained by the service provider. By doing so, the service provider and the user are on the same page in terms of the customized functions, data availability, and outcome visibility. In general, the program instances fall into two categories: map explorer and aggregation charts. In the map explorer, each set of objects (point, line, or polygon) could be visualized in multiple dimensions through the specification of symbol size, color, and time. In addition, the user can apply a specific filter to a specific column and narrow down the analysis scope. Thus, the map explorer enables the user to learn the network status with ease.

On the other hand, an aggregation chart can be generated interactively on the call of session-based requests to provide a standardized representation of MOEs and rapidly identify the feedback such as delay, travel time, speed, etc. Example functions used in various scenarios include the speed analysis based on road link, zonal level origin-and-destination analysis, agent level path evaluation,

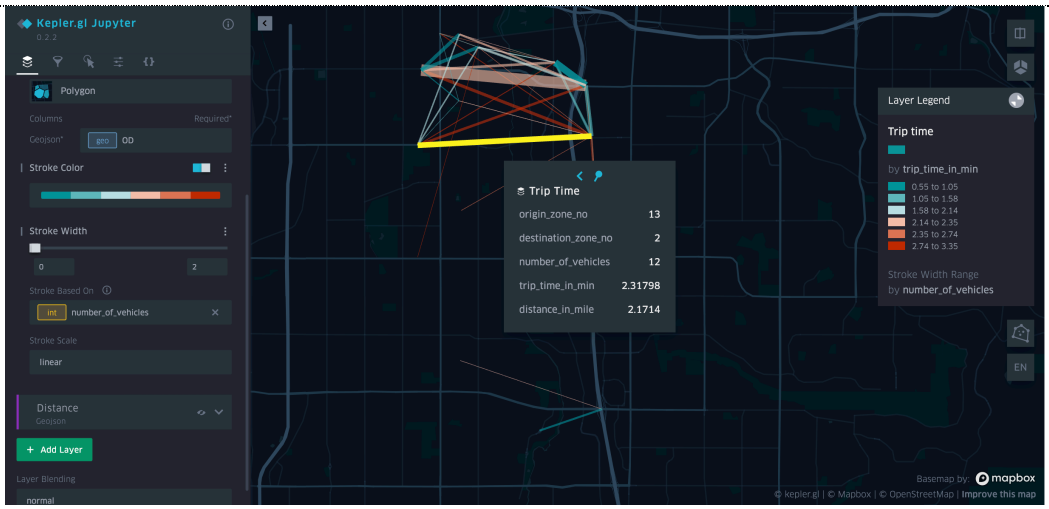
path level spatiotemporal aggregation, system-level temporal statistics. Figure 5.8 shows the screenshots of those example functions.



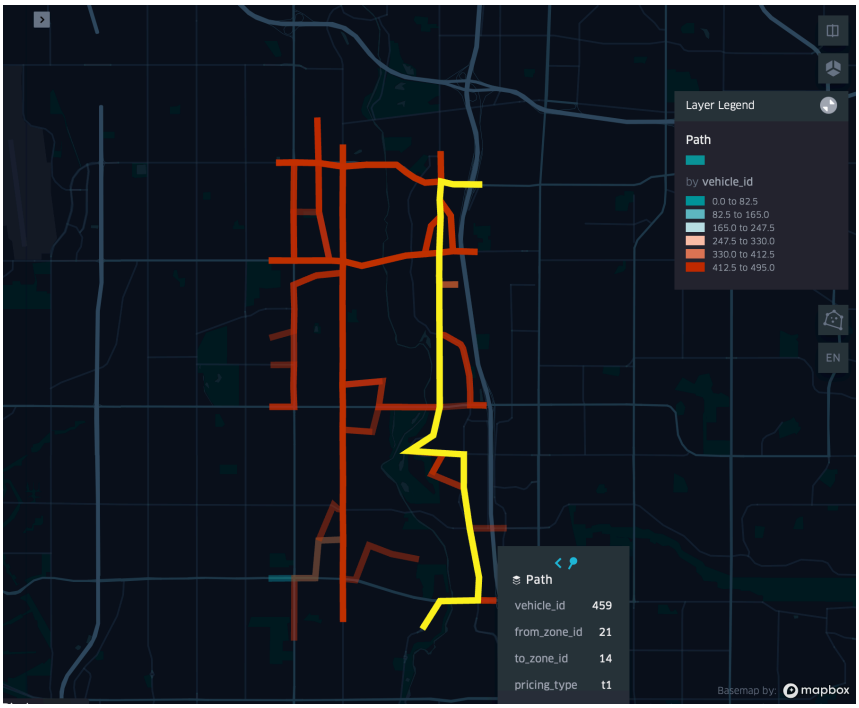
(a) link level MOE—speed



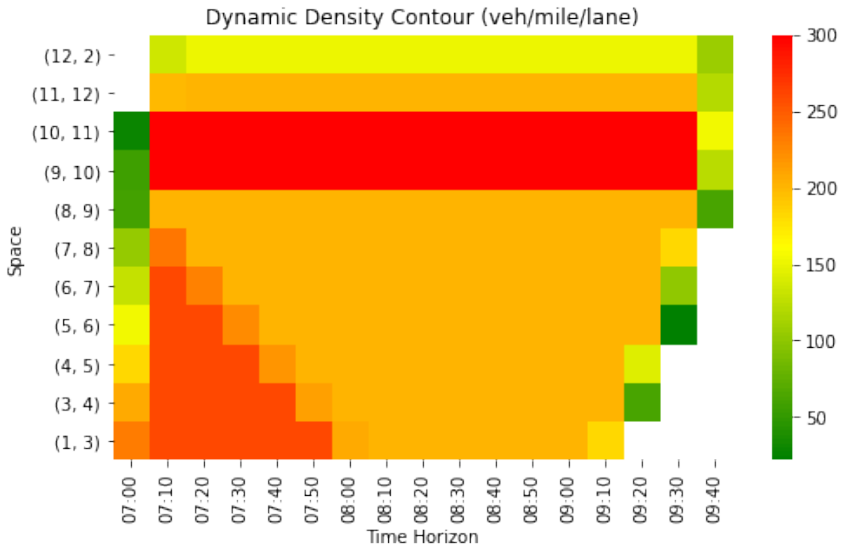
(b) Zonal level MOE—distance



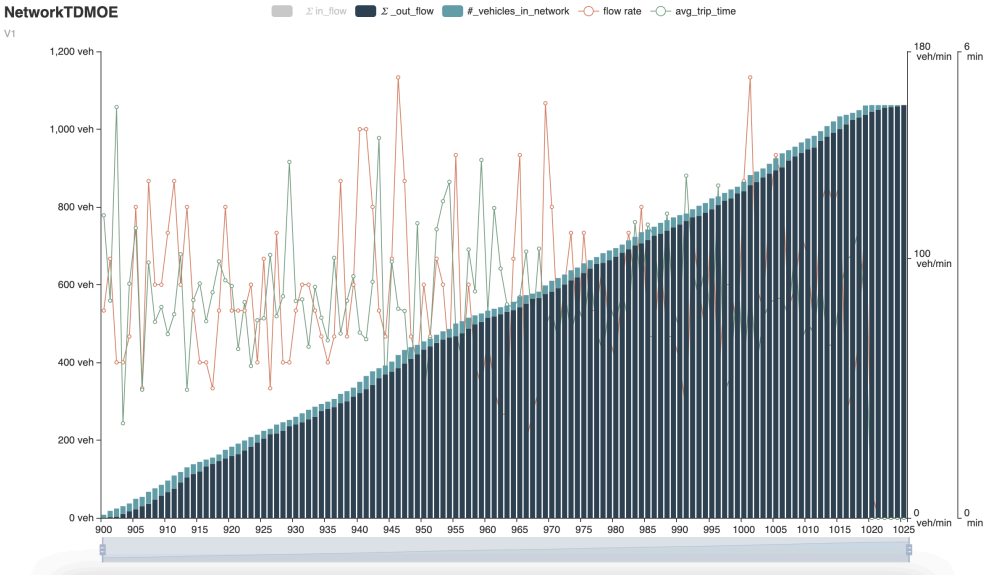
(c) Zonal level MOE—Trip time and number of vehicles



(d) Agent level MOE—Path (zone to zone)



(e) Path level density contour map



(f) Network-level time-dependent MOE

Figure 5.8 The screenshots of a custom analysis

5.3 Social Involvement

As mentioned before, a simple simulation service involves various interactions with stakeholders. A complicated project on the other hand requires more involvement from the transportation simulation community since the service is a highly customized solution adjusted specifically to its customer. This requires the service providers to have professional experience for customization. Professional experience refers to: project experience, engineering judgment, software proficiency, and open communication. Besides experience, the variability in functions or tools in the domain software is essential for a customized scenario. However, correctly setting parameters depends on experience and software capabilities. Lastly, the service cannot be completed by a function alone, but depends on a logically assembled workflow. Therefore, the experience and software usage would be algorithmically represented by the workflow. The flexibility and evolution of the workflow mechanism are the keys to knowledge representation and reproduction.

To cultivate the domain communities, the cloud platform designed a crowdsourcing framework to harness the collective intelligence from three aspects regarding the level of encapsulation. The most widely accepted way to abstract experience from work or communication is through text, pictures, or videos. That information can be organized and formatted as a post in the rich text editor. Then, the creator publishes the post following the subscription and follower design. Here, the creator could be either the service provider or the user. Secondly, for the more capable service providers who can develop and customize tools, they can encapsulate the computational part into code within the online code editor. After the proper configuration of the online virtual environment, the code is ready to be remotely evaluated and represented as a function.

Therefore, a post's scope is beyond the text, pictures, or videos but a new form of a function that can provide insights based on its algorithms and inputs. Finally, as a code subclass, the workflow can also be instantiated and published to a post for social communication and collaborations. Compared to a single function, the workflow is a user case-centered and service provider-generated product, which is more illustrative to its customers and reproducible to the service providers.

The platform helps the creator reach their target audience at different levels through the three types of initialization and publication design. Firstly, for the multimedia/rich-text post, its user is mainly depicted as the people who have a fundamental engineering background. The intentions of this group of people often include the interests of learning new skills, understanding principles or getting insights. Thus, the multimedia post requires fewer efforts from the creator but a higher bar for its target audience. Secondly, as the post has higher abstraction in computational procedures, the post embedded with the program instance has a clear advantage in simplicity since the target audience is not require to have substantial technical training to perform the computation, users simply check the output. Checking the output is much more intuitive than evaluating the methodology. An encapsulated and reproducible program instance would attract more audiences, but it entails a service provider's programming proficiency. Last is the publication of workflow. It asks for not only programming proficiency and engineering experience but also problem-solving ability. It is an integrated and scalable solution supported by a bundle selling strategy from a business viewpoint. Therefore, the target audience will most likely be small, but it would be the primary revenue stream for the qualified service provider and platform.

Depending on the post's estimated value, the creator (service provider or user) has the ability to put a price tag on their post, which creates a channel to earn money from customers by providing domain knowledge or a service. An entirely free-market competition mechanism would work on the platform to balance users' needs and the price of posts. Because the transaction can take place flexibly among the three types of posts at a fine-grained level, the entire simulation community would have more choices in solution and more sustainability in industry development. On the other hand, the cloud platform encourages free posts to share valuable experiences among the community. The underpinning business model would bonus those creators through the content traffic analysis.

The cloud service ensures the technical potential to turn fundamental IT services into "commodity" services (Coronel & Morris, 2017), but the business model also plays an essential part in attracting traffic, nurturing social involvement, and facilitating the transaction of commodity services. Therefore, two business canvasses are presented below to summarize the proposed business model.

By addressing the challenges in meeting proposed value propositions, the platform would create a sustainable service life cycle for the simulation community for all the stakeholders involved. Once the minimum viable product is operating with positive feedback, it will attract more social involvement and nurture the community.

The first canvas is formulated in the view of the cloud computing platform provider. As shown in Figure 5.9, the sections with the pink color background summarize the key partners, activities, resources of the business owner, and the cloud computing platform provider. On the other hand, the green color sections show the customers' side factors of the business cycle, in

which the customer of the proposed platform are the simulation service provider and their end-users. At the bottom of the canvas and painted in gradient color, the two sections represent the cost and revenue stream, which is critical for maintaining the long-term platform operation. In summary, the value propositions in the business model can be stratified by its customer segments. The simulation service provider in the platform would have the advantages in lean service launch, full life cycle support, and scalable service design. Meanwhile, the end-user can enjoy the benefits of service-centered networking, portable tools, low learning cost, pay-as-you-need plan, and quick service evaluation.

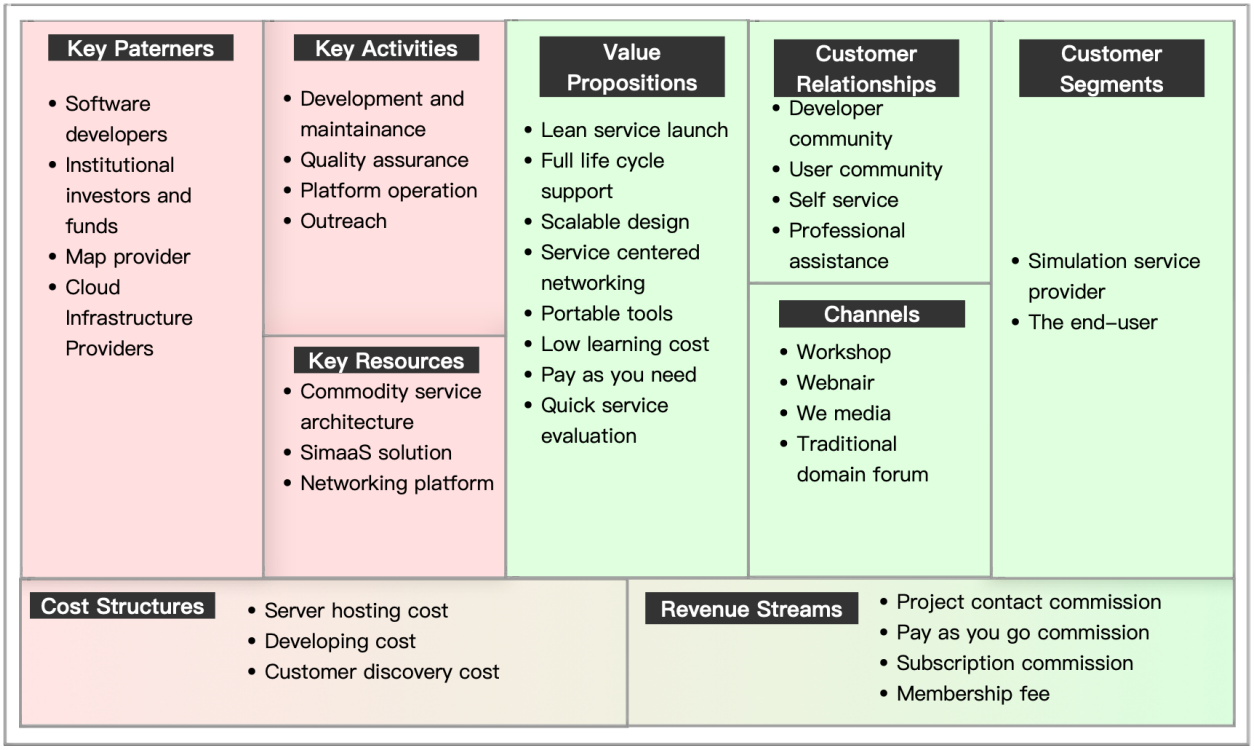


Figure 5.9 The business canvas for the platform provider

Built on top of the platform's business model, the second canvas, shown in Figure 5.10, is designed for the service providers. Here, the upper layer business transpires between the service provider and the end-user. The number and quality of the services deployed and listed on the platform would determine its attractiveness to the end-user and affect the continuing evolvement of the cloud platform. The more advantages the platform can offer to the service provider to facilitate their key activities and provide unique value propositions, the more service providers would join the developer community. However, each service provider needs to carefully weigh the cost and revenue before launching a new service on the platform. If the provider can make the business happen and earn profits by leveraging the underlying features of the cloud platform, a cycle with positive revenue feeds would be formed to support community growth.

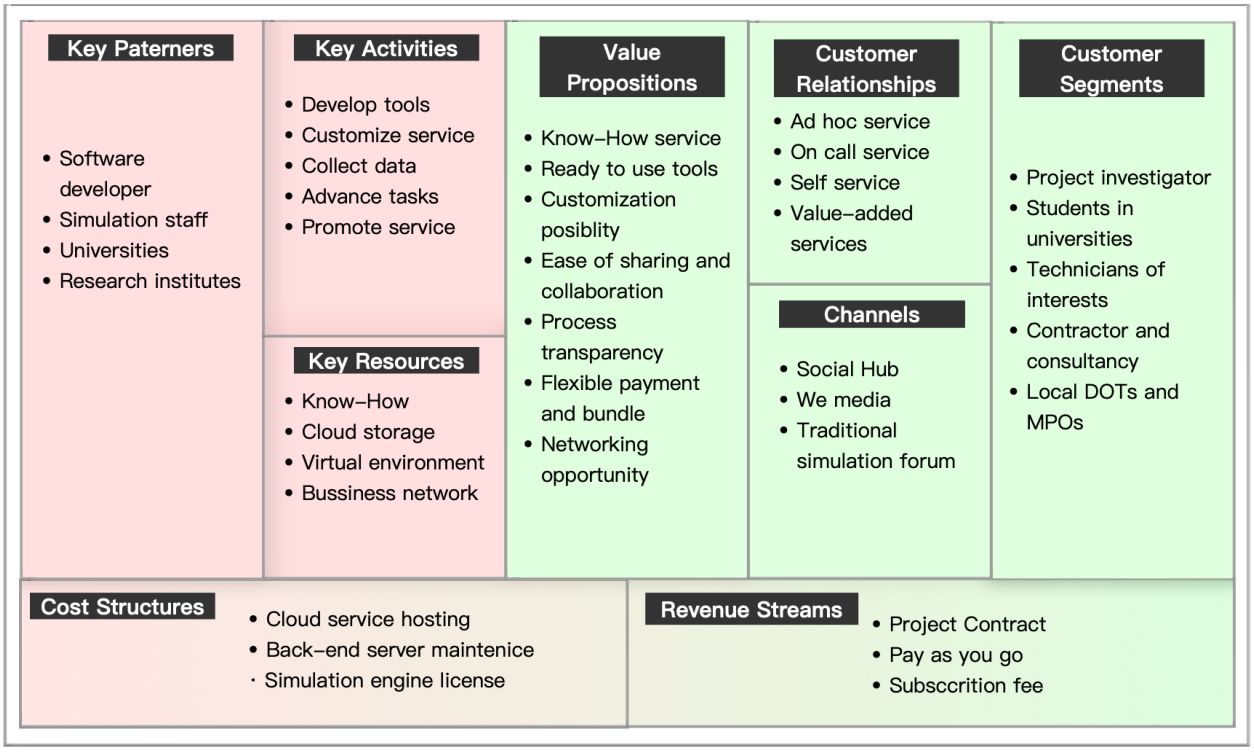


Figure 5.10 The business canvas for the service provider

Chapter 6 Case Study

6.1 Introduction

This chapter presents a case study of the simulation process on the platform, which includes: (1) Simulation service development, (2) Data collection and preparation, (3) Base model modeling and calibration, (4) Alternative analysis (5) Simulation output representation and management. With a minimum manual effort for data entry and calibration, the proposed platform utilizes the developed program instances and assembled workflows to provide the specific scenario analysis. The improvement on both data interoperability and software extensibility brings more benefits to the simulation services while eliminating the barriers between different formats and software functions.

6.2 Service: Real-time LOS evaluation of a signalized intersection under an incident

Before contract negotiations, a service provider should have demonstratable capabilities and have relevant experience in solving similar cases. Specific to this goal, a service provider should have a lightweight editor to develop an incident scenario, a microsimulation tool, and an intersection signal evaluation tool. In contrast, the user is responsible for providing the basic incident description, understanding the optimized control plan, and informing the decision-maker.

The entire workflow is assembled, realized, and powered through the proposed cloud computing platform. The representation of the user interface is established in the Map-Oriented workspace. Moreover, the data pipeline adopted is the combination of the "file instance + data model + service tool + cloud engine." For this example described in this chapter, the file instance

is an excel template based on the intersection data scheme. The data model is a JSON representation of the control section in the intersection data scheme. The cloud engine is the CORSIM executable in TSIS 6.0 deployed in a Windows Tower server. The service tool is a published program instance on the platform, which is run by users and is responsible for processing the data inputs (excel template and JSON supplement), translating to the input format of the CORSIM engine, communicating with the simulation server, retrieving simulation output files, parsing and post-processing the outputs, and generating the evaluation dashboard.

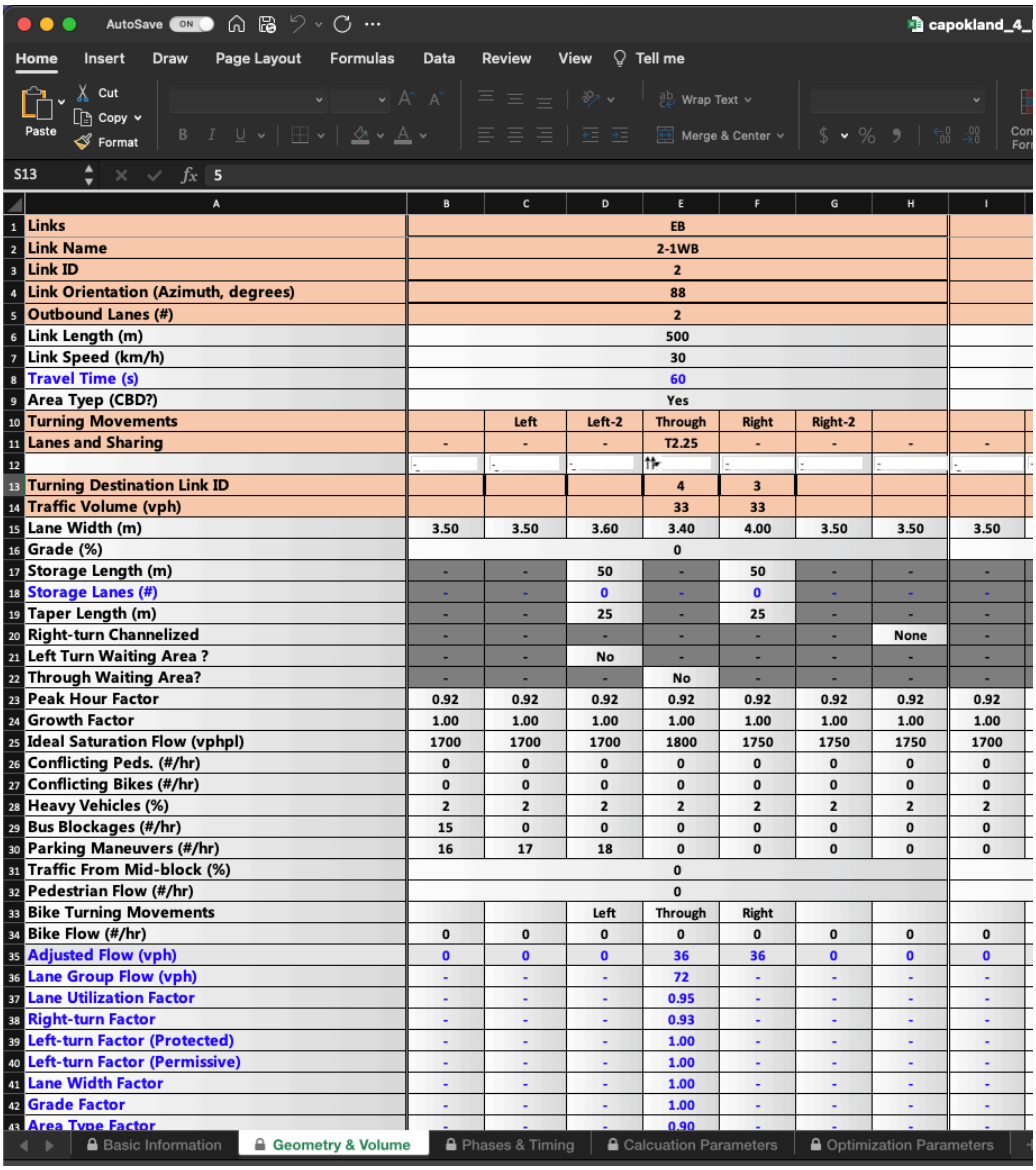
The service provider guides the user to fill in the correct information through UI design, documentation, and instant service in a Map-Oriented workspace. Then, the user only needs to enter the workspace, follow the workflow, run the service, obtain scalable outputs, and manage them properly in cloud or local storage.

6.2.1 Case background

The first service is the evaluation of a four-leg intersection near the UW-Milwaukee campus area. The intersection data was collected in the Excel template. The optimized signal plan, average peak-hour volume, and simulated LOS are shown in Figure 6.1. The Excel template is a two-dimensional representation of the data scheme described in Chapter 4. This calculation-embedded spreadsheet contains five tabs: Basic information, Geometry, Volume, Phases and Timing, Calculation Parameters, and Optimization Parameters.

6.2.2 Service scenario

In this case, a signal plan evaluation is conducted between a service provider and user pair. Therefore, they share the expected outcomes such as the modelled intersection spreadsheet, the dashboard incorporated HCM method and Simulation Engine, and the simulation input file.



AutoSave ON — capokland_4_

Home | Insert | Draw | Page Layout | Formulas | Data | Review | View | Tell me

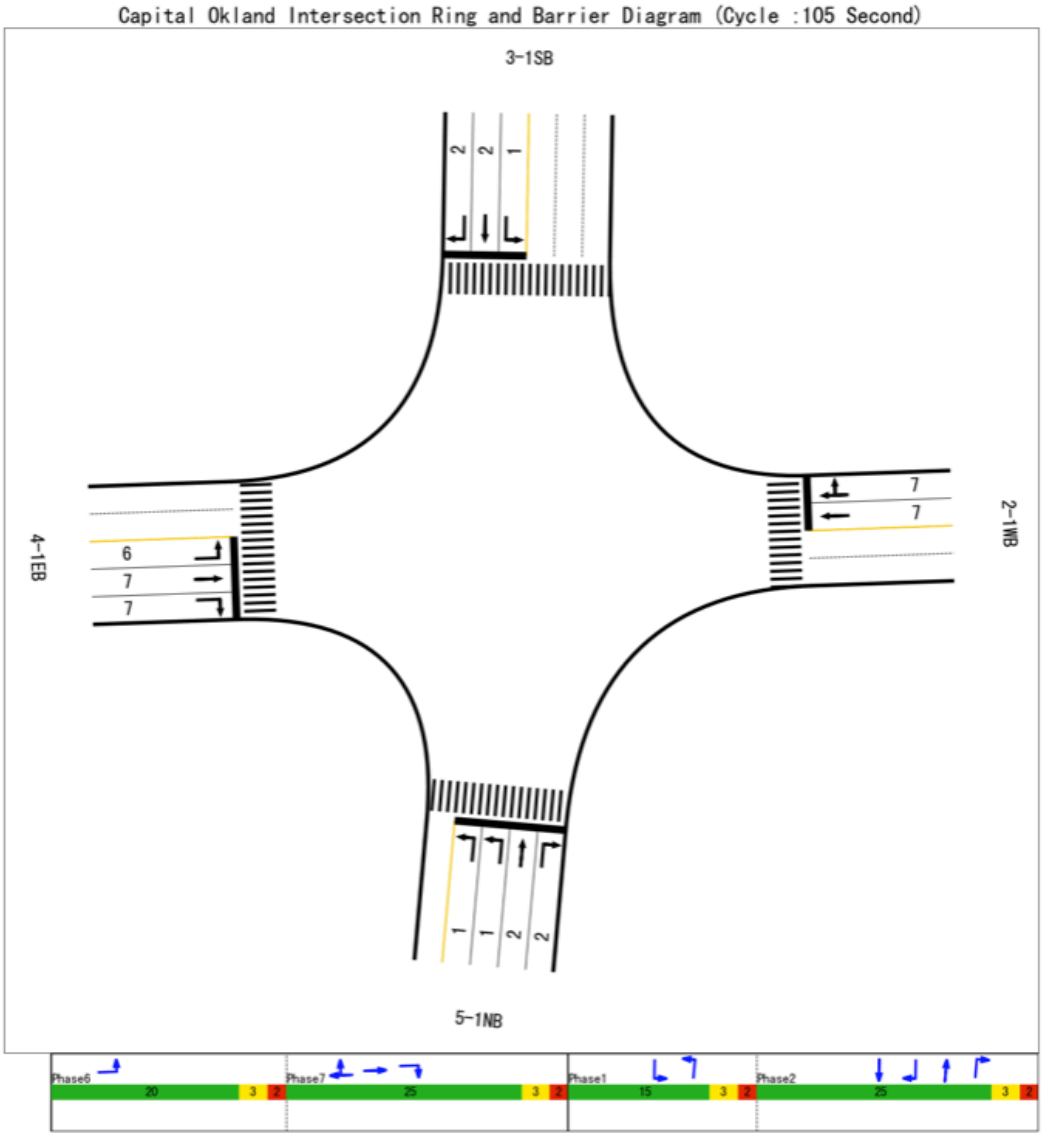
Cut | Copy | Format | Paste | Wrap Text | Merge & Center

S13 | 5

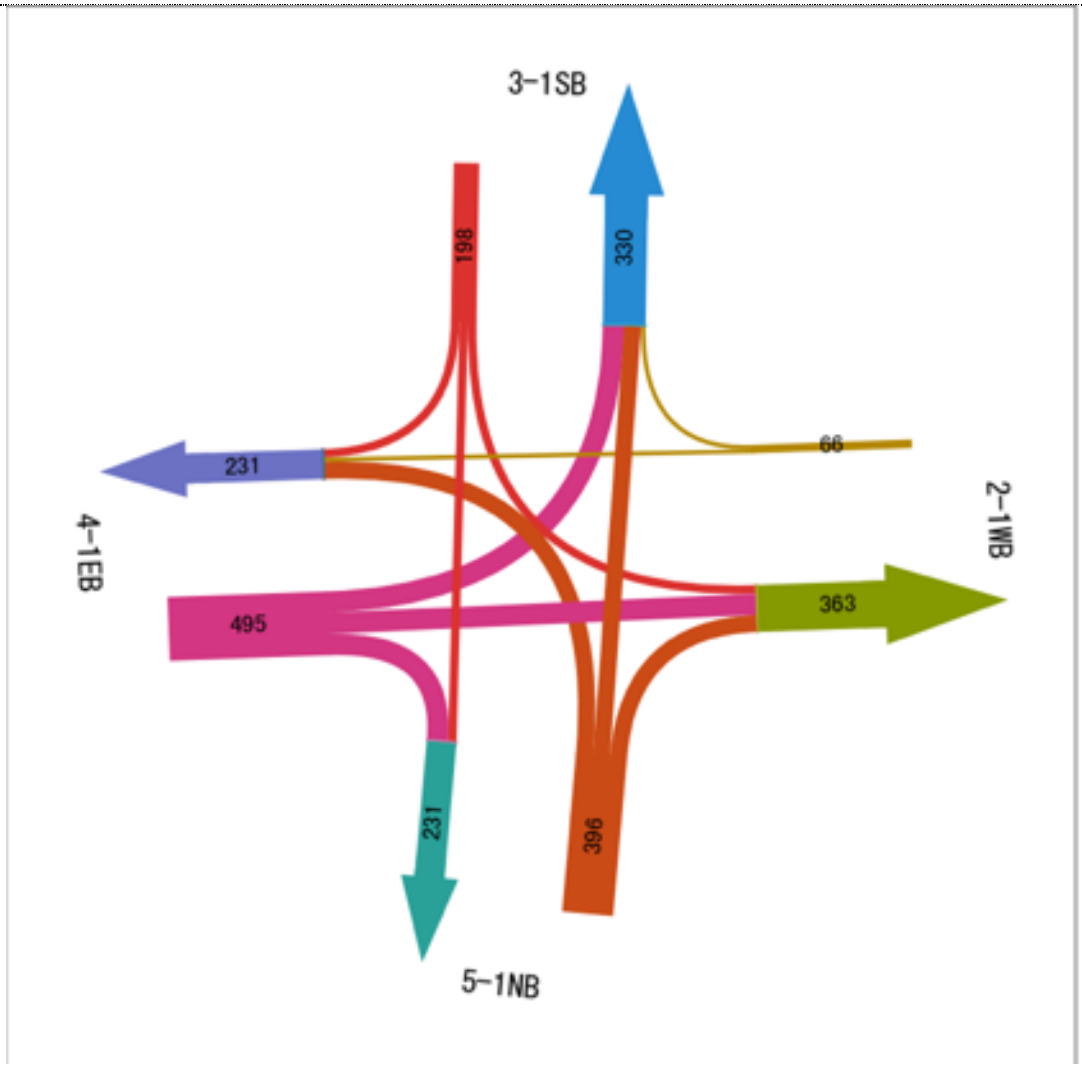
	A	B	C	D	E	F	G	H	I
1	Links				EB				
2	Link Name				2-1WB				
3	Link ID				2				
4	Link Orientation (Azimuth, degrees)				88				
5	Outbound Lanes (#)				2				
6	Link Length (m)				500				
7	Link Speed (km/h)				30				
8	Travel Time (s)				60				
9	Area Tyep (CBD?)				Yes				
10	Turning Movements		Left	Left-2	Through	Right	Right-2		
11	Lanes and Sharing	-	-	-	T2.25	-	-	-	-
12									
13	Turning Destination Link ID				4	3			
14	Traffic Volume (vph)				33	33			
15	Lane Width (m)	3.50	3.50	3.60	3.40	4.00	3.50	3.50	3.50
16	Grade (%)				0				
17	Storage Length (m)	-	-	50	-	50	-	-	-
18	Storage Lanes (#)	-	-	0	-	0	-	-	-
19	Taper Length (m)	-	-	25	-	25	-	-	-
20	Right-turn Channelized	-	-	-	-	-	-	None	-
21	Left Turn Waiting Area ?	-	-	No	-	-	-	-	-
22	Through Waiting Area?	-	-	-	No	-	-	-	-
23	Peak Hour Factor	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
24	Growth Factor	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
25	Ideal Saturation Flow (vphpl)	1700	1700	1700	1800	1750	1750	1750	1700
26	Conflicting Peds. (#/hr)	0	0	0	0	0	0	0	0
27	Conflicting Bikes (#/hr)	0	0	0	0	0	0	0	0
28	Heavy Vehicles (%)	2	2	2	2	2	2	2	2
29	Bus Blockages (#/hr)	15	0	0	0	0	0	0	0
30	Parking Maneuvers (#/hr)	16	17	18	0	0	0	0	0
31	Traffic From Mid-block (%)				0				
32	Pedestrian Flow (#/hr)				0				
33	Bike Turning Movements			Left	Through	Right			
34	Bike Flow (#/hr)	0	0	0	0	0	0	0	0
35	Adjusted Flow (vph)	0	0	0	36	36	0	0	0
36	Lane Group Flow (vph)	-	-	-	72	-	-	-	-
37	Lane Utilization Factor	-	-	-	0.95	-	-	-	-
38	Right-turn Factor	-	-	-	0.93	-	-	-	-
39	Left-turn Factor (Protected)	-	-	-	1.00	-	-	-	-
40	Left-turn Factor (Permissive)	-	-	-	1.00	-	-	-	-
41	Lane Width Factor	-	-	-	1.00	-	-	-	-
42	Grade Factor	-	-	-	1.00	-	-	-	-
43	Area Type Factor	-	-	-	0.90	-	-	-	-

Basic Information | Geometry & Volume | Phases & Timing | Calcuation Parameters | Optimization Parameters

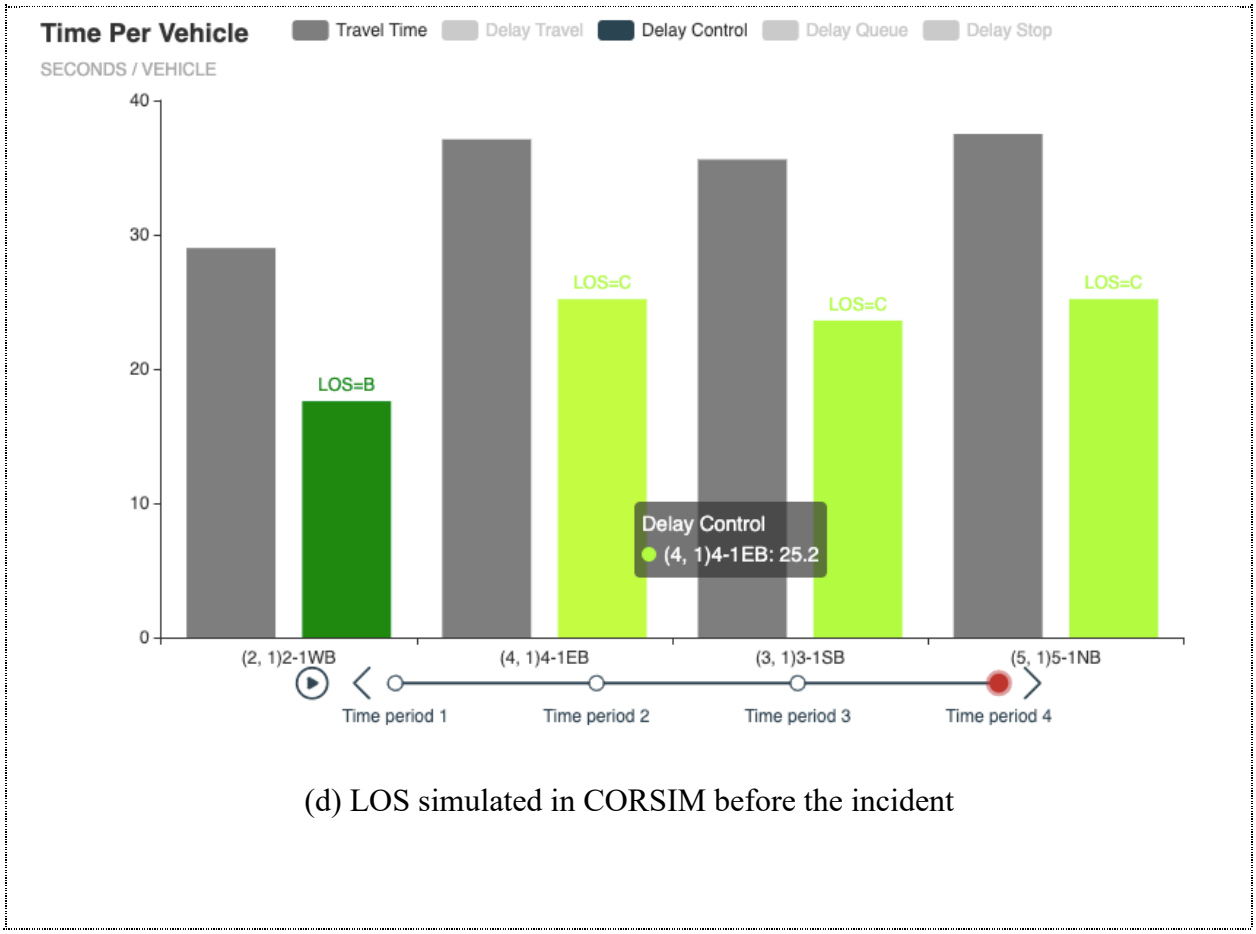
(a) Excel template for intersection data collection



(b) Original control plan and channelization



(c) Volume of vehicles in each movement



(d) LOS simulated in CORSIM before the incident

Figure 6.1 Simulation-based LOS evaluation before the incident

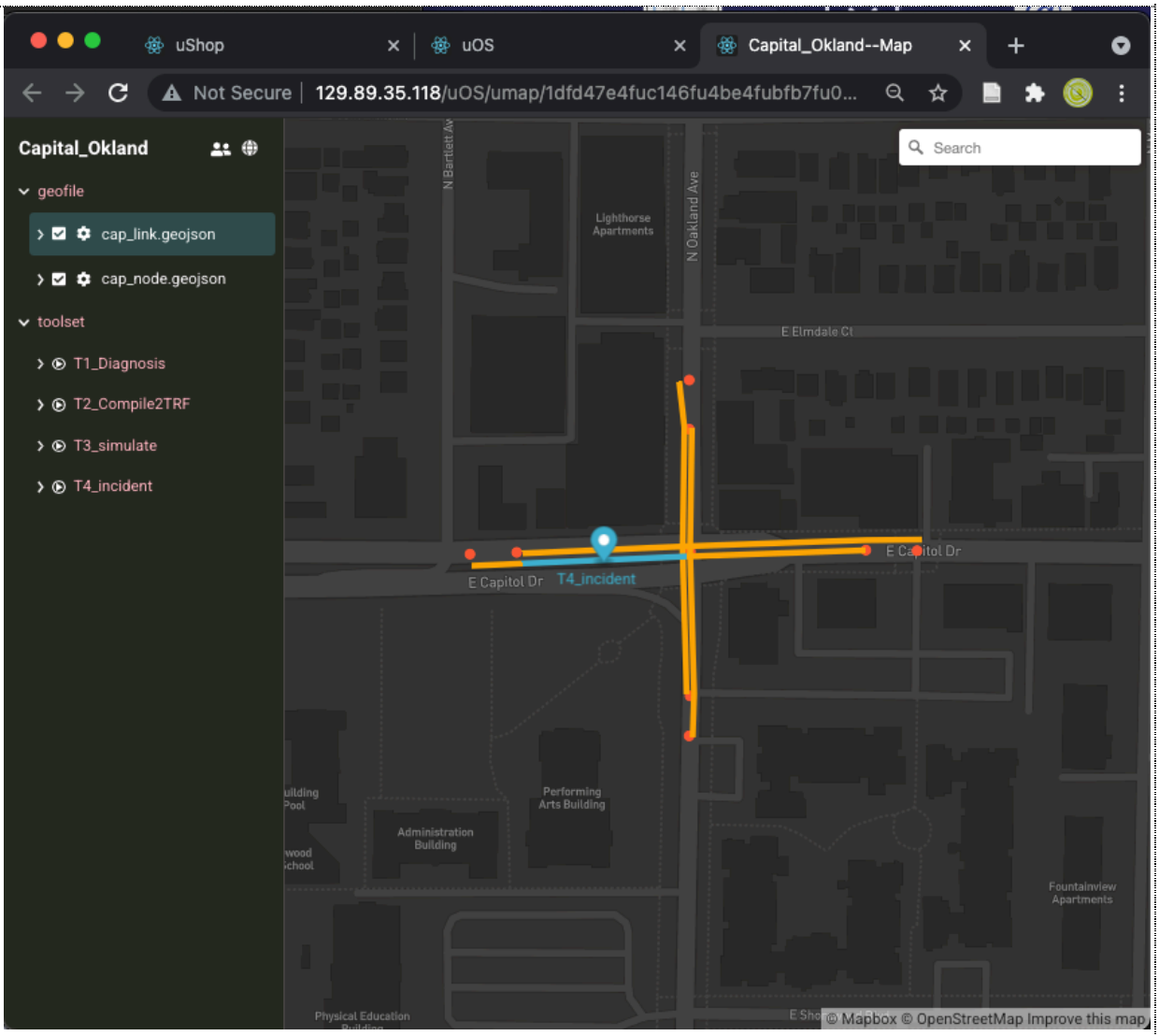
6.3.3 Service life cycle

One of the most frequent intersection base modeling services used is the incident impact evaluation. The user, usually the traffic operator in a state DOT, wants to evaluate the impact of an incident in an intersection using the microsimulation method. The service provider created a Map-Oriented interface, shown in Figure 6.2 (a), to provide a succinct and exploratory interface for the traffic operator.

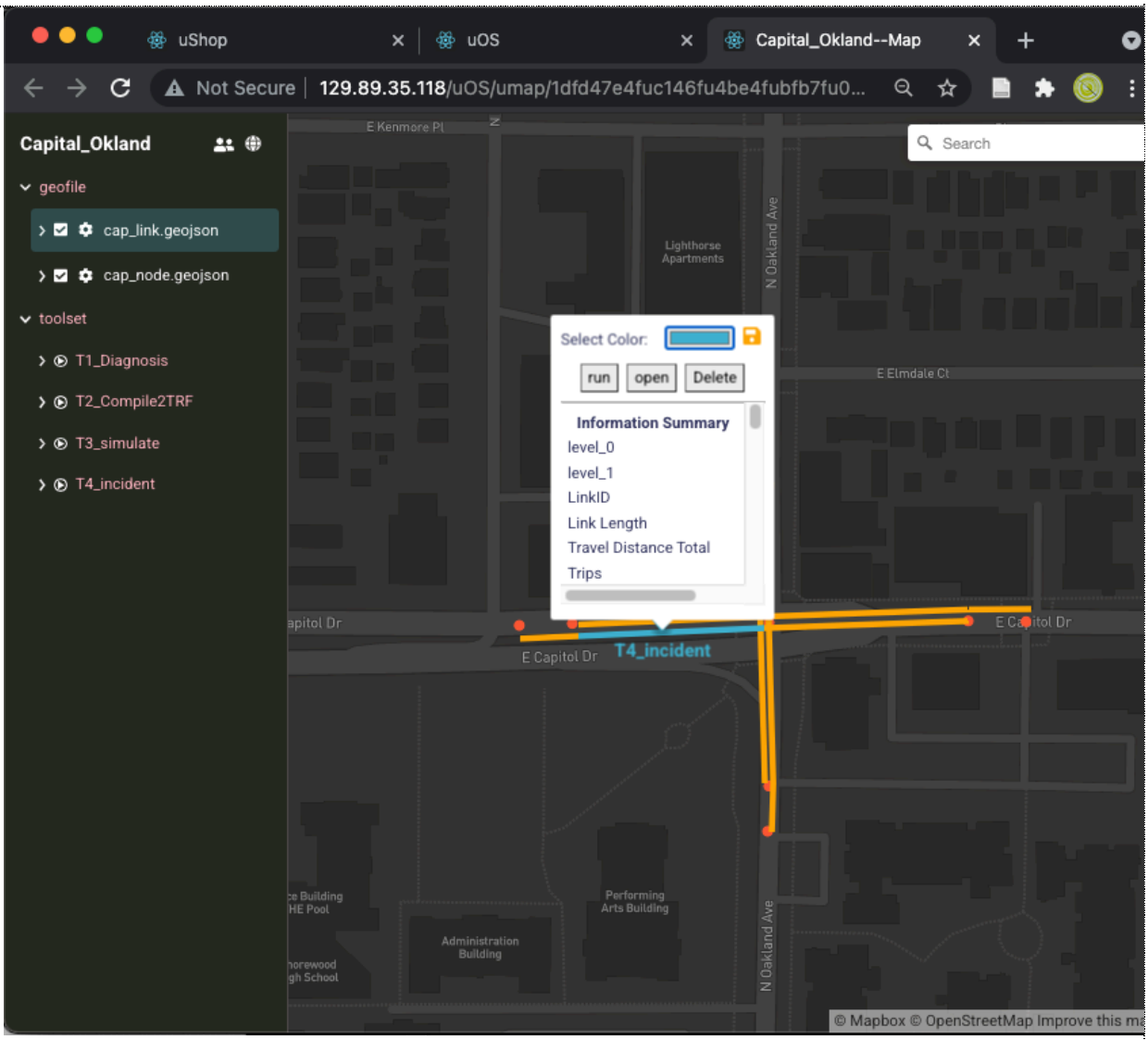
As the left panel shows, the tasks in the workflow as named using the letter T represent a service instance and a number behind to represent the task sequence. In Figure 6.2 (a), four

functions are assembled under the folder “toolset,” from T1 to T3 are the tasks used in the pre-conducted service. T4 refers to the operations that the user needs to interact with to access the incident impact. On the right-hand side, the map section illustrates a georeferenced implication of an intersection, in which the eastbound approach link, labeled as (4,1), is bound with the service tool (T4_incident). In summary, the workspace is a type of operation desktop, which enable the service provider to assemble the user-purchased tools and present a customized function in a map-oriented manner.

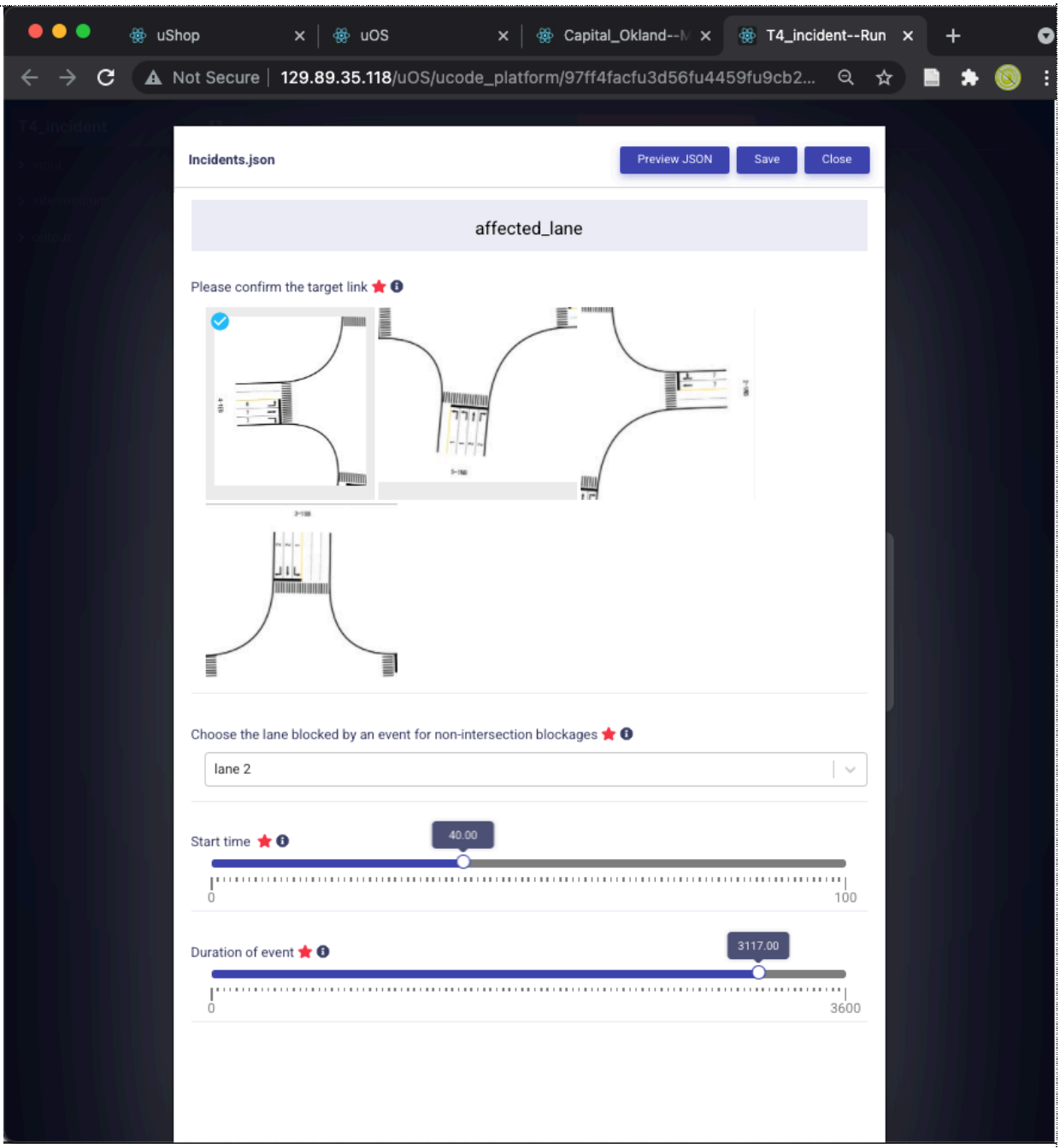
When a user wants to simulate an incident in the link (4,1), they can click the blue segment and check the options in the pop-up dialog shown in Figure 6.2 (b). If the “Run” button is clicked, the browser will pop up a new tab, and the user can follow the steps to specify the parameters defined by the service provider. A frequent type of parameter collection is completed by the data model through the visualizer in the platform. As shown in Figure 6.2 (c), the user needs to answer four questions to identify an incident. Upon completing the data model, the service tool is ready to evaluate, and the real-time response appears in the right panel, the terminal window, illustrated in Figure 6.2 (d). In addition, the user can manage the input and output files in the left folder panel. Thus the generated dashboard can be moved to any other place in the Cloud Storage. It is worth noting that, before the service tool is published, the service provider can set visibility and permission to the folders and files contained in the developed tool. This mechanism, depicted in Figure 6.2 (e), ensures simplicity, reliability of the program instance, and the ability of the service provider to modify the system.



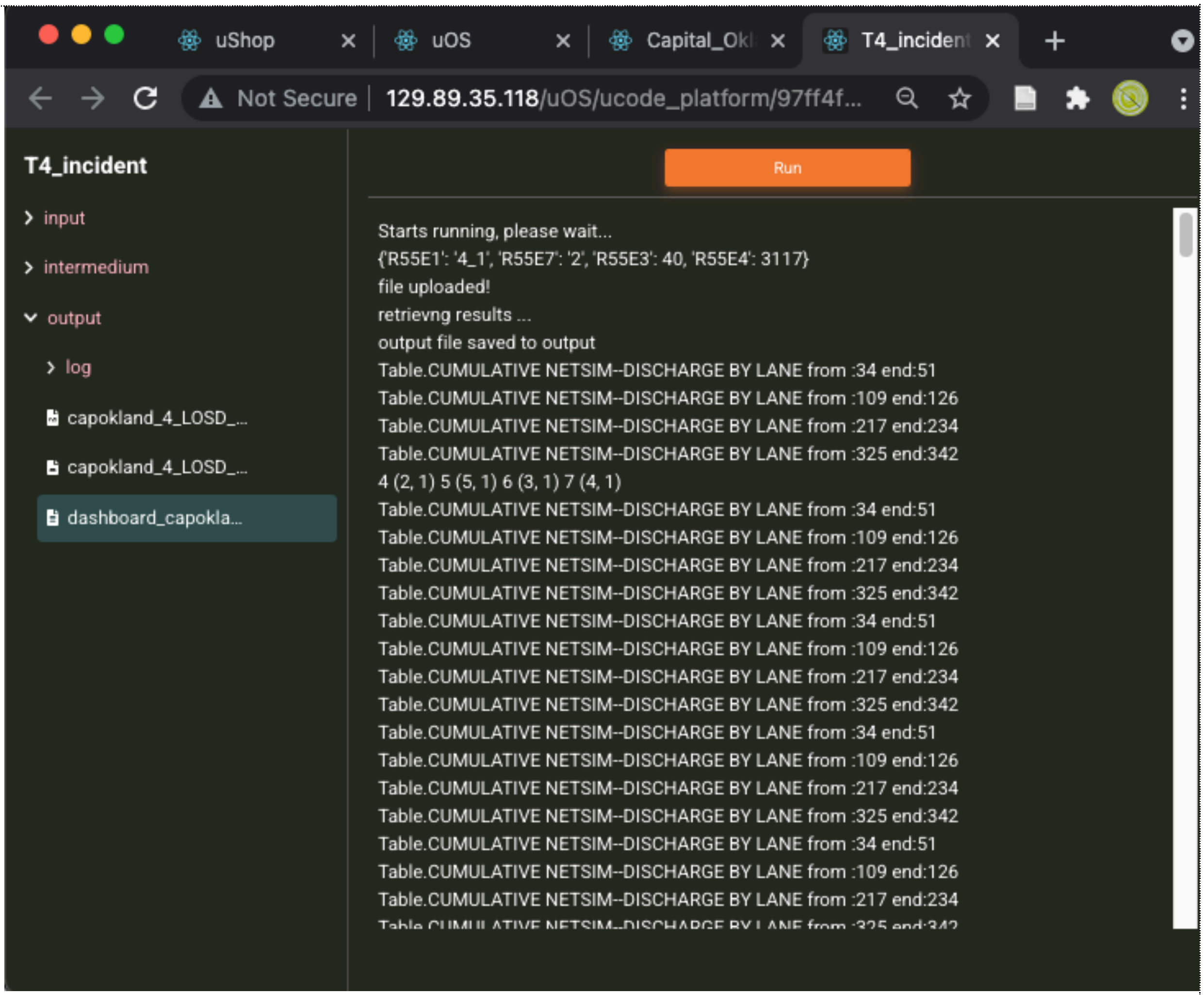
(a) The customized map-oriented workspace for incident simulation (User's view)



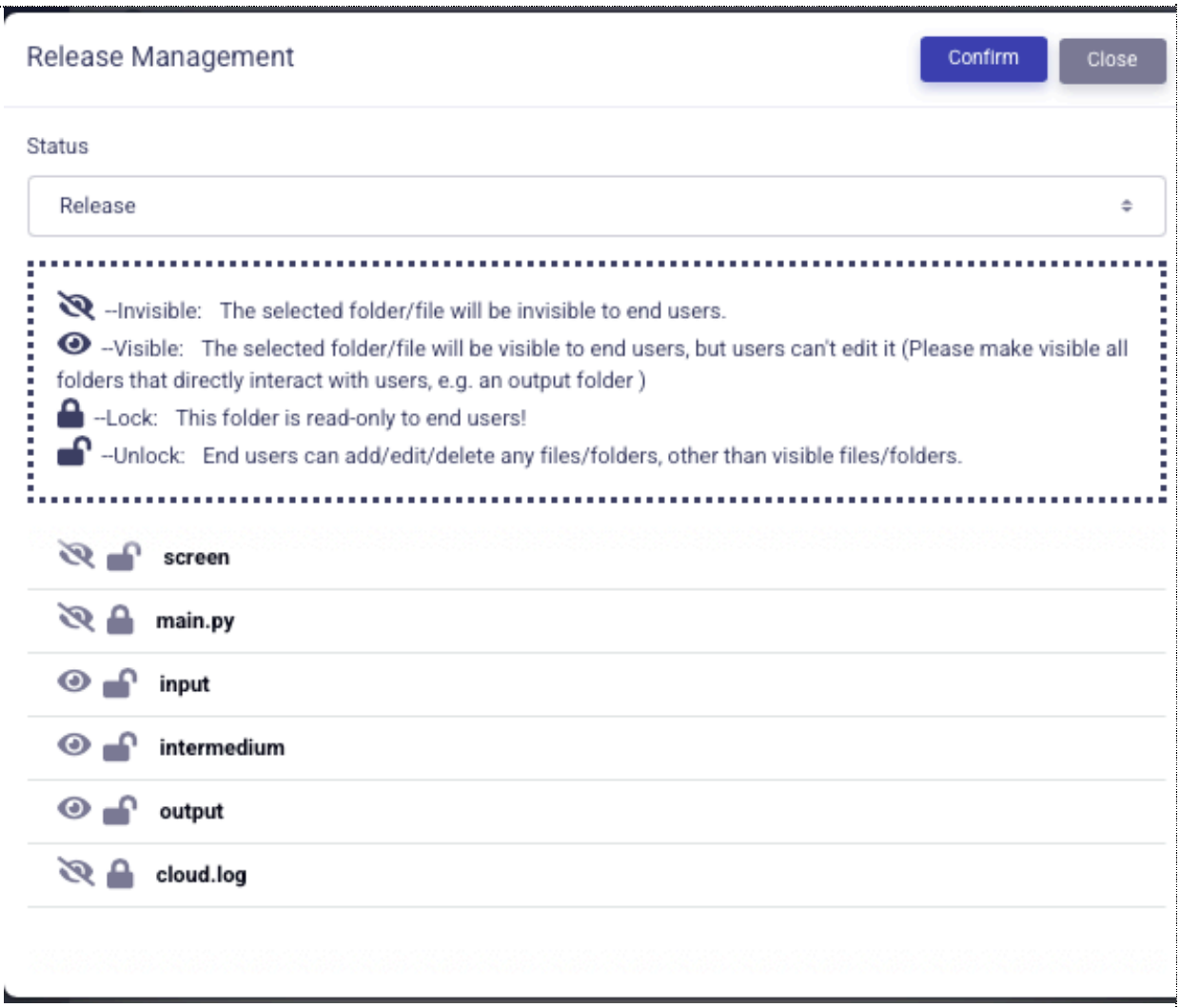
(b) The tool embedded in the selected approach link (User's view)



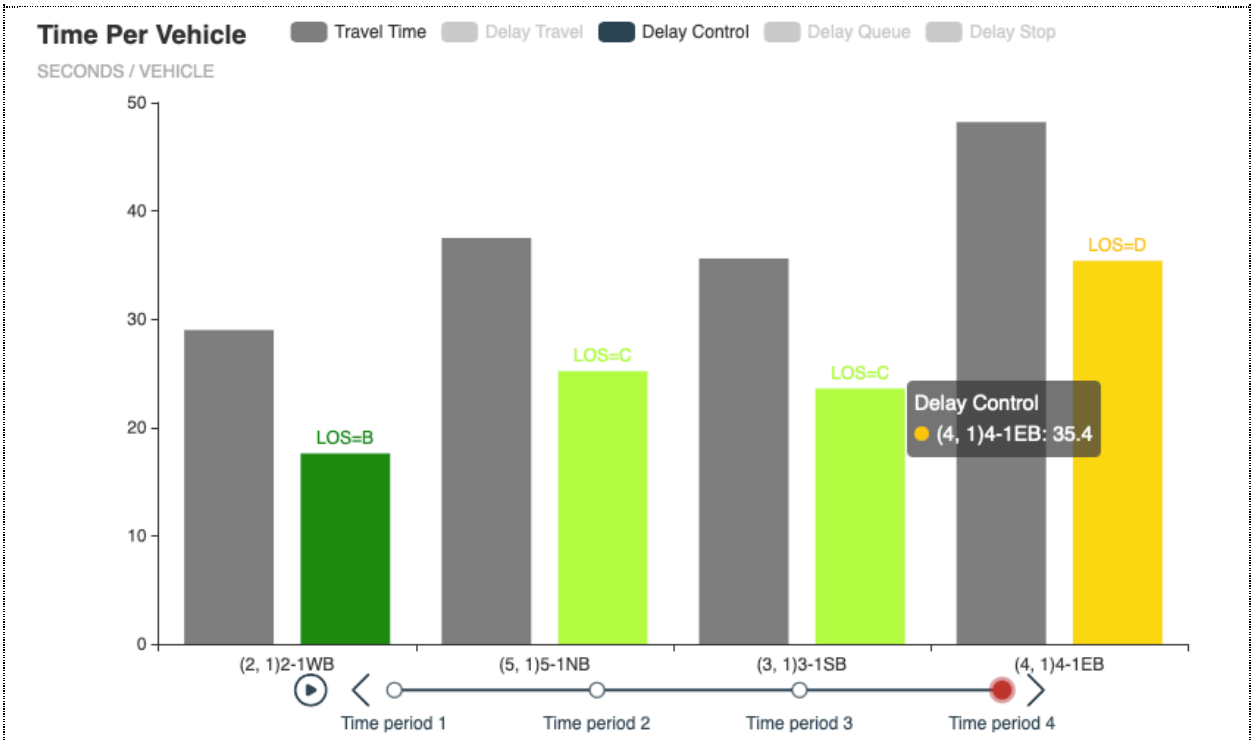
(c) The data model used to identify the attributes of the incident (User's view)



(d) The terminal feedback and the output management (User's view)



(e) Visibility and locking mechanism of the incident simulation tool (Service provider's view)



(f) LOS simulated in CORSIM after an incident

Figure 6.2 Simulation-based LOS evaluation after the incident

Comparing Figure 6.2 (f) with Figure 6.1 (d), it can be clearly identified from the chart that, when an incident occurs on the middle lane of the approach link, which starts in 40 seconds from the beginning and lasts for 3117 seconds in the designed 3600 simulation cycle, the LOS service is degraded from the original level C to level D. The average delay for each vehicle increases from 25.2 seconds to 35.4 seconds, which is a perceivable delay for any traveler.

To minimize the perceived delay under this circumstance, the user, who can be an operator in the department of transportation, can request a signal optimization service from the service provider. To some extent, the service provider's capability would determine the extensibility and functionality of the simulation service.

6.3 Service: a streamlined set of online planning tools for public evaluation and assessment

446.3.1 Case background

Dr. Xuesong (Simon) Zhou leads the ASU Transportation AI Lab team developed a series of planning tools following the GMNS format using Python. A public GitHub repository named "integrated_modeling_GMNS" synthesized a continuous flow of GMNS-AMS modeling. The continuous workflow is listed in **Table 6.1,** with description, input and output files specified for each step, and software packages.

Table 6.1 The continuous workflow in GMNS-AMS Traffic Network Modeling (X. Zhou, 2021)

Step	Description	Software	Input Files	Output Files
0	OSM data download	Open Street Map (OSM)	- -	Map.osm
1	Convert OSM data to GMNS	OSM2GMNS	Map.osm	Node.csv, Link.csv, poi.csv
2	Convert GTFS data to GMNS	GTFS2GMNS	Open transit data GTFS	Node.csv, Link.csv, poi.csv
3	Expand macroscopic network data to micro, meso	net2cell	Node.csv, Link.csv	Meso and micro-networks in node.csv and link.csv
4	Zone-to-zone travel demand	grid2demand	Node.csv, Link.csv, Poi.csv, Poi_trip_rate.csv	Demand.csv, Zone.csv, Accessibility.csv, Input_agent.csv
5	Traffic signal for timing	Data2timing (In development), Sigma-X	Node.csv, Link.csv, Movement.csv	Timing.csv
6	Capacity estimation and VDF calibration	Data2vdf	link_performance.csv	vdf_table.csv, link.csv

7	AMS simulation	A/BStreet, Path2GMNS DTALite	Demand.csv, Node.csv, Link.csv, Input_agent.csv	Agent.csv, Link_performance.csv
8	OD Matrix estimation	Data2demand	link_performance.csv Agent.csv	Demand.csv,
9	Visualization	QGIS, NeXTA	Node.csv, Link.csv, Movement.csv, Zone.csv, Demand.csv	- -

6.3.2 Service scenario

The tools listed in Table 6.1 win a wide reputation among the community in the traffic planning sector. One of the popular packages, OSM2GMNS, had over 2000 downloads within two months after its initial release on November 5, 2020. In the meantime, its relevant documentation, tutorials, and demonstration video, distributed in all social media channels account for over 20,000 views in total. Even though many developers and modelers have the capability to investigate the code and model, the effort to turn those who know how to download and deploy the package into those who know how to use the package is non-trivial. In addition, the number of potential users who are interested in using the function but with limited programming experience is almost tenfold the number of developers. However, the challenge of converting an interested user into an experienced developer is almost insurmountable and typically not economical.

To move these code-on-demand style applications a step closer to potential users, without increasing the effort spent on deployment and promotion, skilled developers can leverage the proposed cloud computing platform to deliver the remote evaluation style service.

6.3.3 Service life cycle

Beginning with the independent processing stage, a service provider (the creator of the package or a skilled developer with permission) can collect feedback from interested users via various channels and logically convert that customer's needs into scripts. As is depicted in Figure 6.3, a service provider can structure, test, and develop a script of a GMNS visual analytic tool in a code-oriented workspace underpinned by Cloud Storage. The developer can set the virtual environment, folder structure, and default executable script within the editor interface. Once a developer completes testing, they can bring this tool into the surrogate interaction stage.

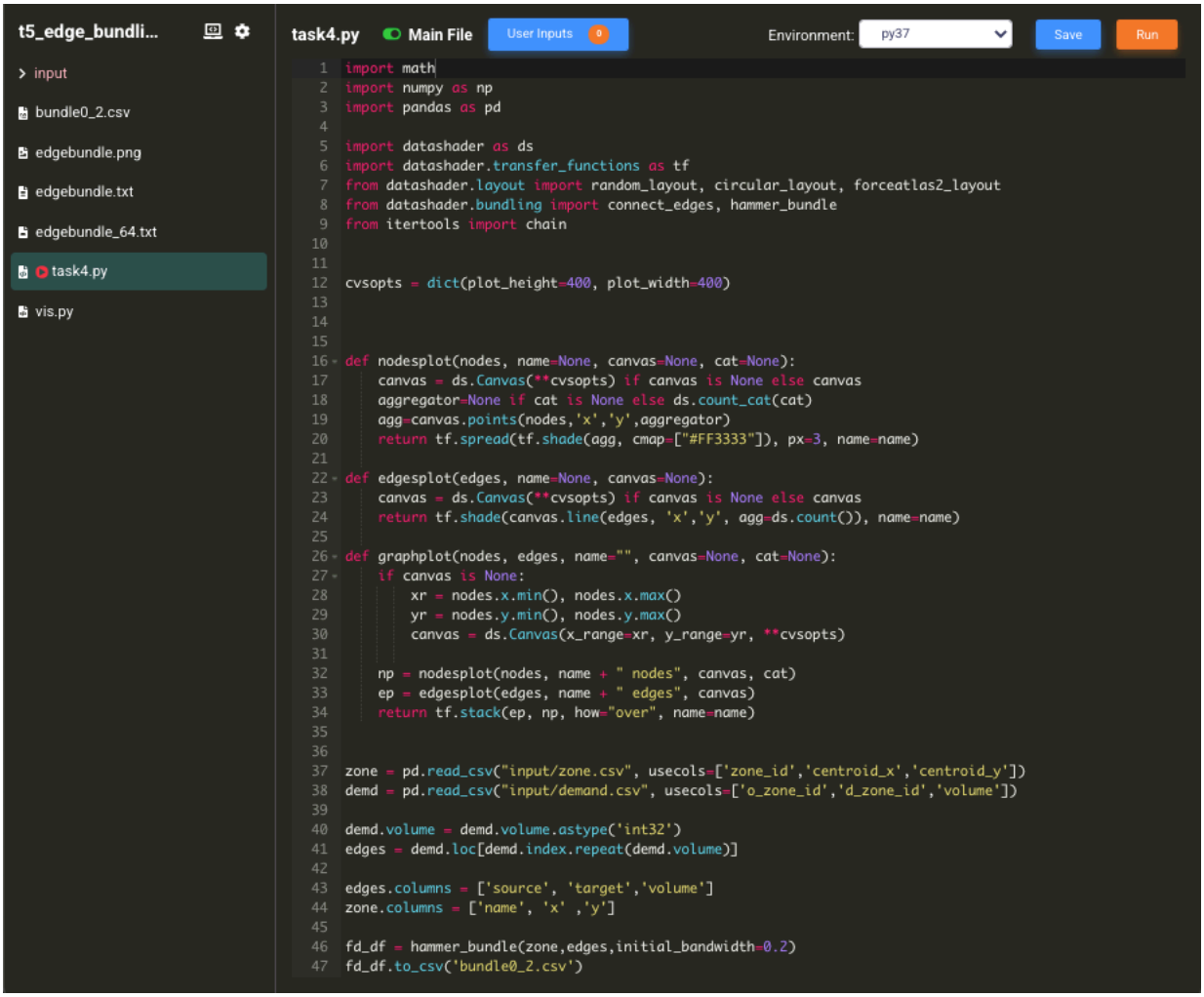


Figure 6.3 A sample tool tested in the code editor on the Cloud Storage

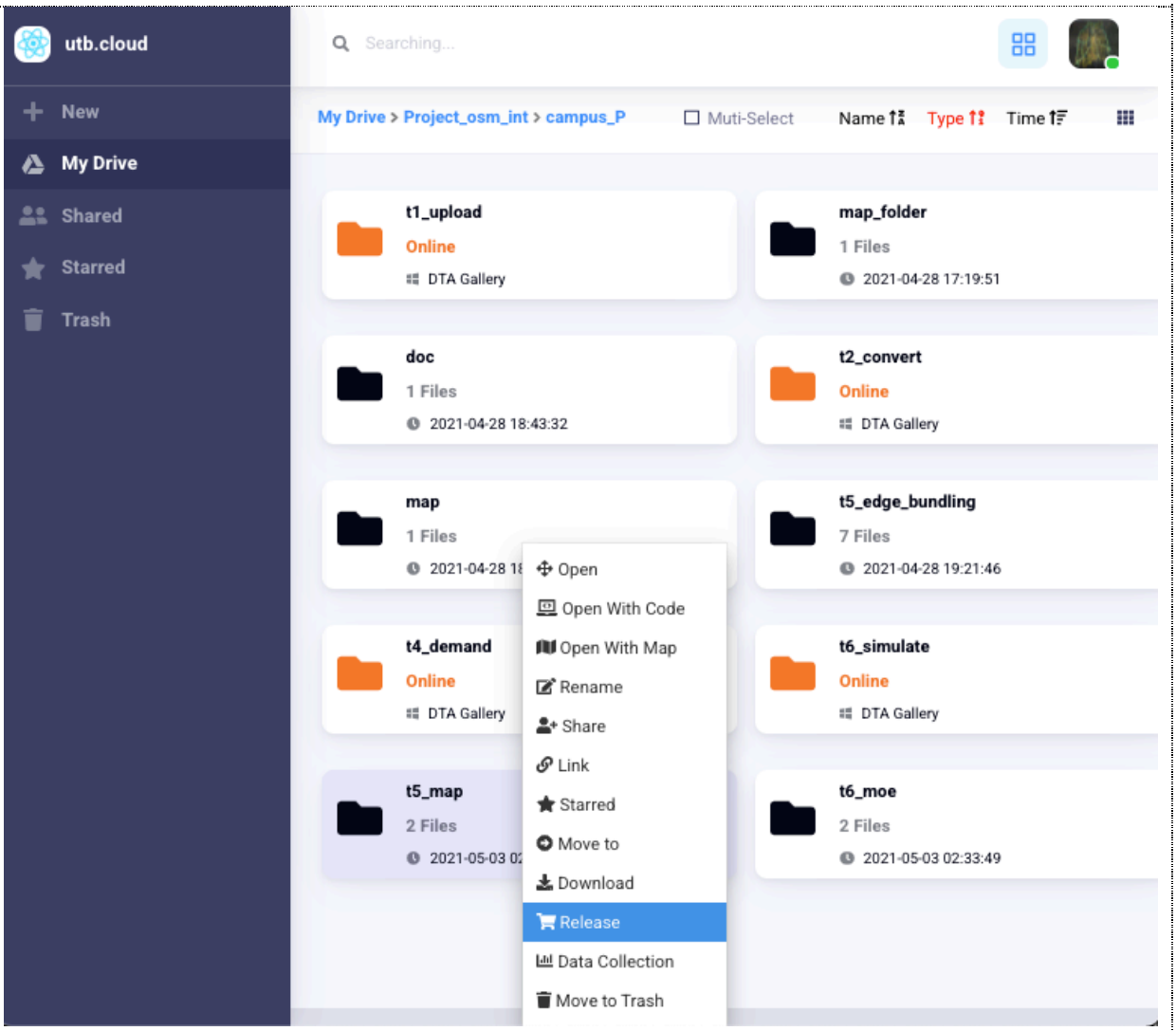
In the surrogate interaction stage, a service provider is responsible for tailoring a users' needs into a workflow, which is: model each task as a tool, order the task sequence, and release the tools and sequence into the Social Hub. As seen from a service developer view in Figure 6.3 (a), Cloud Storage enables the developer to perform release management, then automatically put on a new symbol to represent the status of a published tool. All tools can be managed and integrated into a post on the Social Hub, which is a place to attract more users and solicit potential new features. Nevertheless, each published tool is a public link about a specific roadway intersection that any user can access, but the link is merely runnable to site users and the intersection data cannot be edited from this run-only link.

Therefore, a registered user attracted by the run-only online tools can purchase the service for their own needs. Figure 6.3 (b) provides a user's view after clicking on the link and finishing the purchase process. Within this description page of a specific tool, a user can purchase, read documentation, write comments, and rate the tool. More importantly, the cloud platform precisely connects a locally runnable but domain-specific tool to its users and brings the user to a runnable place with high readiness. If the user has a particular need that cannot be met by the existing published tool, they can find the service provider through the account listed in the "contributors" tab.

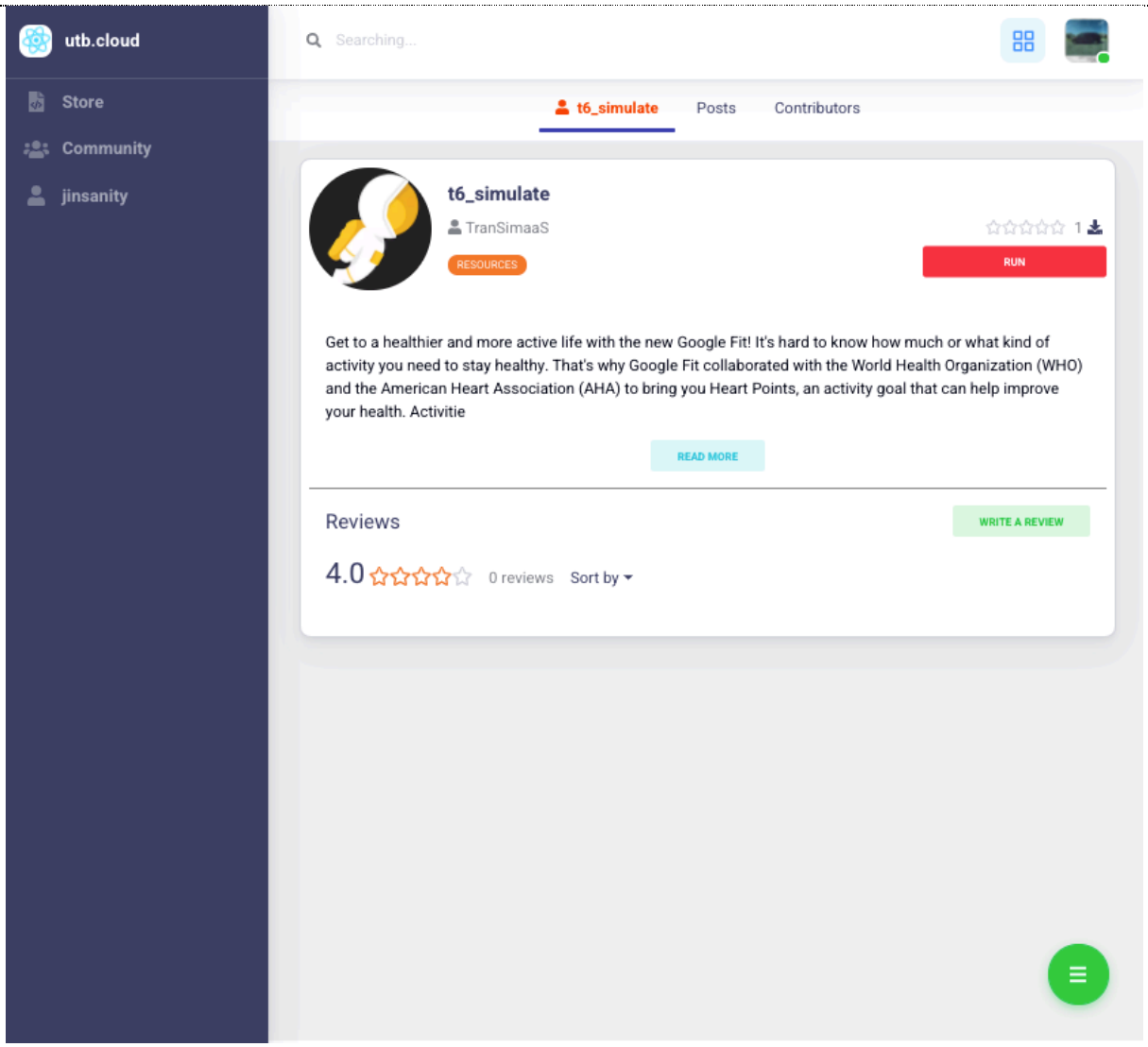
A direct interaction happens after the connection between a user and a service provider has been established. As has been discussed in Chapter 5, multiple forms of support can be provided, such as: pure service, mixed service, or quasi-manufacturing service. A rule of thumb is that the

service provider needs to weigh the costs and benefits of its business model before agreeing to support a tool.

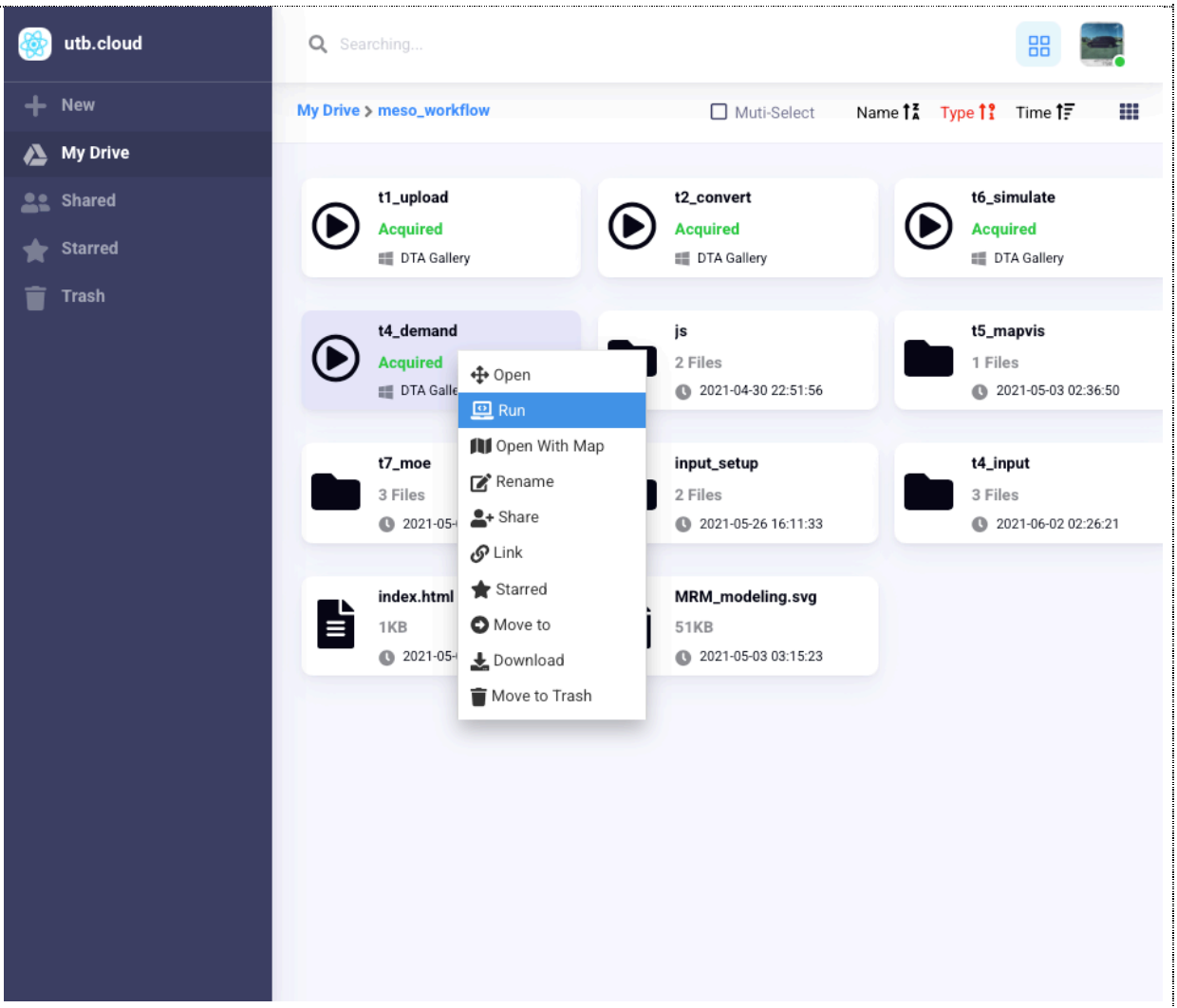
After purchasing a tool, a user will have a program instance in their own Cloud Storage space, enabling them to perform an evaluation through the back-end system architecture described in Chapter 3. As seen in Figure 6.3 (c), a user will see service instances with an execution symbol and action menus. If a user clicks the “run” button, the browser opens the page shown in Figure 6.3 (d). The service page opens in a folder structure for management purposes, and has an interactive panel on the right-hand side. Clicking “run” executes the predefined process of the cloud service, which includes a user preference dialog (the data modal) and a prompt to upload the required input files. A user checks the output files on the file management panel or they can check a runtime error in the pop-up terminal on the left.



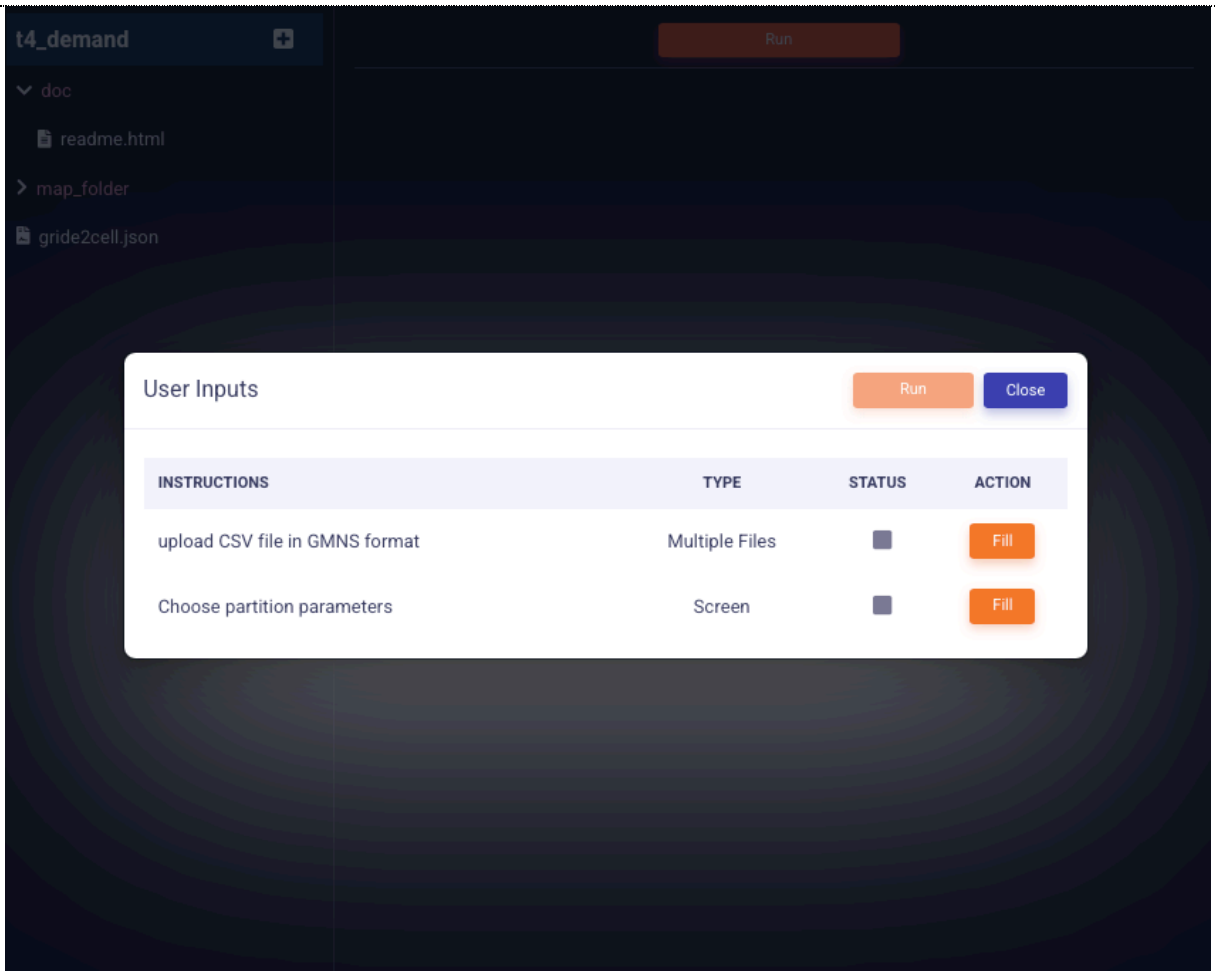
(a) Published program instances from the service provider's view



(b) User's view at the post of a tool on the Social Hub



(c) User's view at its Cloud Storage Space



(d) User's view at the actions dialog before a service run

Figure 6.4 The key steps in the surrogate Stage

The screenshots shown in Figure 6.4 illustrate a proof-of-concept of a typical online tool at multiple stages in both the service provider view and the user view. The purpose of each online tool on the cloud platform is to improve usability and the serviceability of the task itself, rather than replace the implementation style or network-based architectural style. To better instruct a user on the inherence and logical connection of tasks in a service workflow, the service provider can

use a directed cycle diagram to specify the input and output connections visually. As shown in Figure 6.5, the service provider helps the user build a customized service workflow for the GMNS service and has embedded the service tool link in each colored block. Thus, a GMNS service is tailored to the user's needs in the operational desktop as described in Chapter 3. The user can click a blue-colored block to load a dialog modal, a window that forces an interaction. The dialog modal follows the graphic illustration in Figure 6.6.

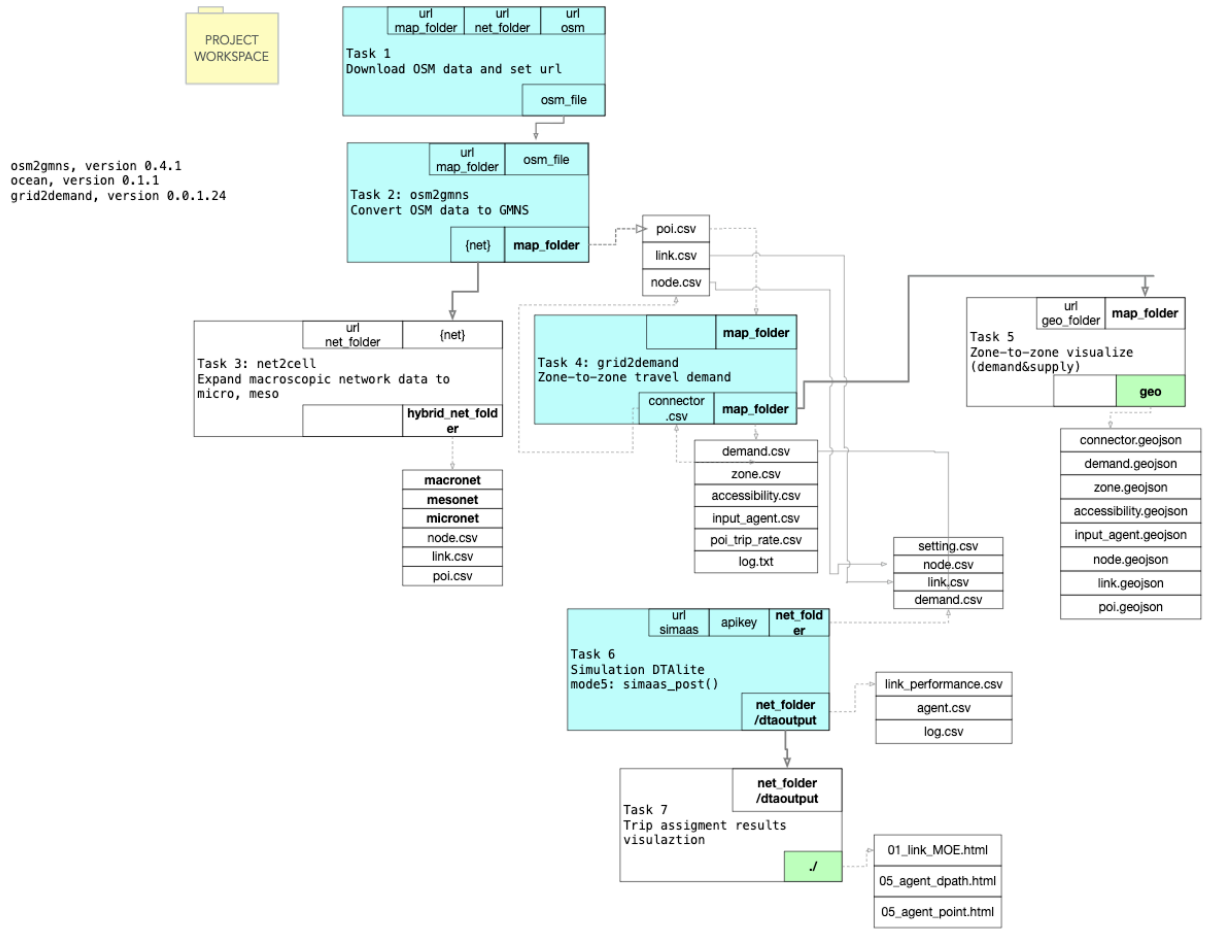


Figure 6.5 An interactive workflow designed for the GMNS service

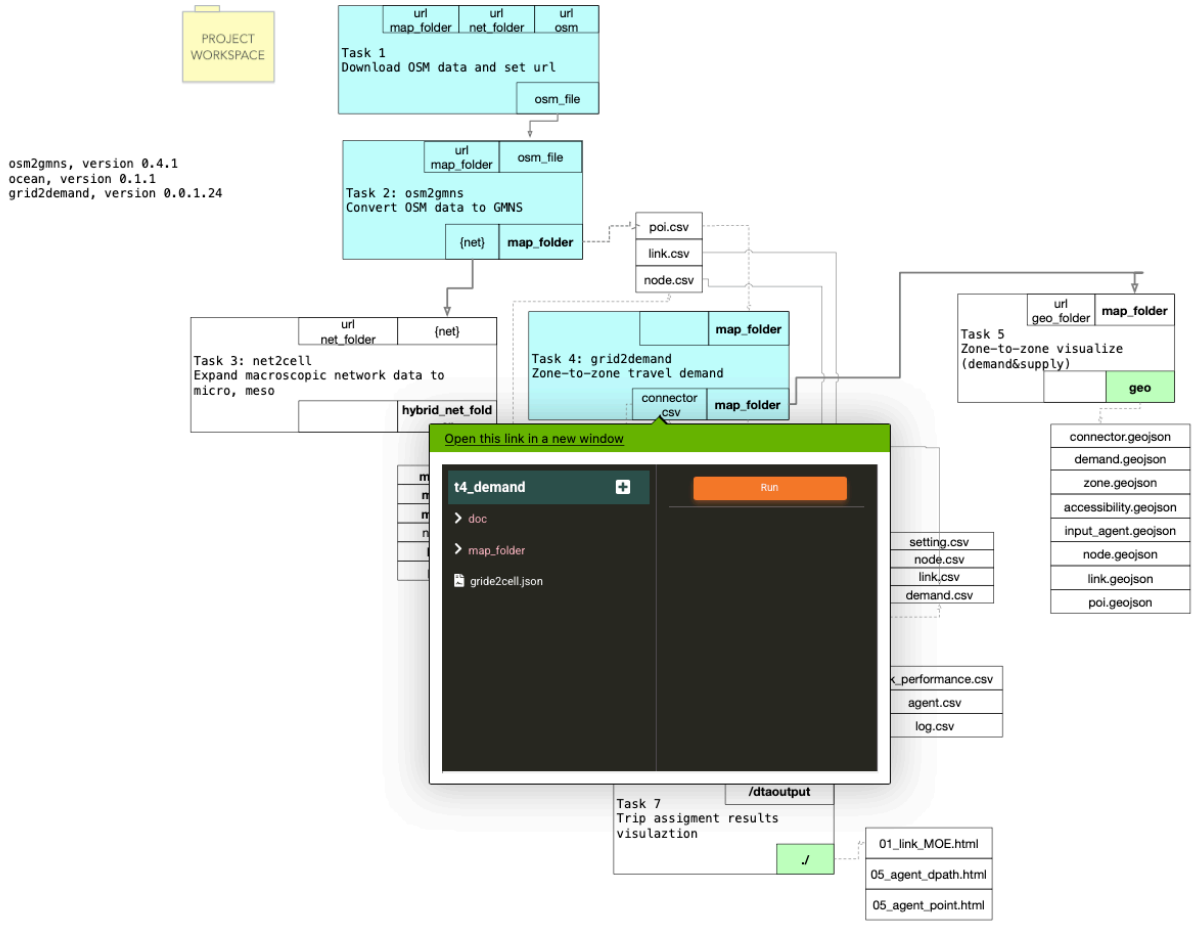


Figure 6.6 The dialog modal of an active task in the workflow

6.4 Simulation Engine Task Scheduler

A person or entity who owns a valid simulation engine resource on a server can be referred to as the simulation provider. In the designed case study, key parameters and assumptions are given as follows:

- Total number of simulation requests in queue: 5
- Number of available simulation engines: 3
- Number of Window workstations or desktops: 3

Table 6.2 demonstrates the arrival pattern of a sequence of simulation requests. Upon the arrival of each request, the cloud middleware will assign a request number, and record its arrival time and destination engine. Then, the estimator in cloud middleware will give approximate run time of each request on various engine and server, where the time estimation is summarized in Table 6.3. Before moving to the assignment task, the middleware also needs to contain the engine availability information like Table 6.4 shows, where 1 stand for the engine is installed in the target sever. With all the information be collected and computed, the MIP model will generate an assignment plan shown as Table 6.5.

Table 6.2 The profile for the queuing simulation requests

Request in queue No.	Arrival time (Sec)	Destination engine*
1	0	1
2	4	2
3	5	1
4	5	1
5	14	3

Notation: * engines 1,2,3 represent the engine of the TransModeler, CORSIM, and VISSIM, respectively.

Table 6.3 The estimated time for the coming task at different engines and servers

Job request towards Engine (id, engine*)	Time estimation (secs)		
	Server 1	Server 2	Server 3
1,1	22	53	81
2,1	27	88	91

3,1	230	659	819
4,1	22	47	66
5,1	377	805	1017
1,2	34	76	137
2,2	74	162	228
3,2	593	1016	1607
4,2	59	86	161
5,2	1139	2230	3343
1,3	50	126	190
2,3	100	191	257
3,3	720	1397	2303
4,3	58	171	166
5,3	1292	2591	4750

Table 6.4 The specification of the available servers

Sever / Engine	Server 1	Sever 2	Sever 3
TransModeler	1	0	0
CORSIM	1	0	1
VISSIM	0	1	0

Table 6.5 The optimized the task schedule and waiting time

Request in queue No.	Destination engine*	Processed in Server	Arrival time (Secs)	Job start time (Secs)	Processing time (Secs)	Waiting time (Secs)
1	1	Server 1	0	0	22	0
2	2	Server 3	2	2	228	0
3	1	Server 1	5	22	230	17
4	1	Server 1	5	252	252	247
5	3	Server 2	14	14	2591	0

In summary, the minimized waiting time of this case is 264 seconds in total. In the case of five queuing requests, The MIP problem contains 151 constraints, 137 variables, and 369 nonzero values, and the optimized result is solved using the CPLEX engine within 430 milliseconds. The computation speed is responsive enough for real-time assignment used in cloud middleware.

Chapter 7 Conclusion and Future Work

7.1 Conclusions

This dissertation developed a service framework for simulation-based studies and tasks in transportation. These tasks support transportation planning needs, such as evaluating a new turn lane, and transportation operation needs, such as signal timing. The entire framework is built on top of a cloud computing architecture to facilitate all stages in a simulation life cycle. By leveraging the features and functions of a cloud computing platform, the capability of the simulation provider can be flexibly and efficiently represented in the form of the service tool. In addition, a user can acquire, evaluate, and experiment with existing online services without having to buy the service.

In a simulation service, there are two significant players, the simulation provider and the user, both of which are defined in Chapter 3. The main objective of designing a simulation platform is to solve the pain points for both the simulation provider and its users. The principal method adopted for drawing up the platform is the Model-Driven approach. By considering the activity context of the general service provider and user, key features, primary components, and functional requirements are specified to guide the initial design of the platform.

From a service provider's point of view, four principal components are developed in a component-based approach to: 1) meet data collection requirements, 2) translate data into appropriate formats, 3) manage and monitor simulations, and 4) handle errors. The flexible combination of the data containers and program instances can collect and process many small and quick simulation needs. A session-based server-side mechanism was designed to call on functions if users need a more comprehensive GUI and imbedded functions. In addition, the simulation

engine is a server wrapped with RESTful APIs to receive, initiate, monitor, and manage ongoing simulation requests. Moreover, cloud middleware is responsible for recording, estimating the simulation requests, and managing the server pool's status. A MIP-based task scheduler is modeled to dispatch the queuing requests to the available servers at minimal waiting cost. Last, a three-phase-error handling mechanism was implemented to help developers and users identify problems and move on quickly.

The design of this service framework reduces barriers that exist in the technical implementation of online simulators. As described in Chapter 5, cloud computing service activities and management as well as techniques for support are presented. Three stages of service interactions are described from the viewpoint of the platform customers, which are both the service provider and the user. The value propositions that the platform can bring to its customers have been discussed. Moreover, three scalable dashboard mechanisms were designed to effectively manage scenarios and visually illustrate MOEs. A business canvas of the platform was created to facilitate the simulation service and social involvement.

Chapter 6 presents two case studies to illustrate: the practical need, the development of a service, the role of the platform, and the concept of Simulation as a Service in the field of Transportation Engineering. Example projects that a start-up service provider would likely encounter are presented and run through the Simulation as a Service approach. In addition, considering future demand, an optimized task scheduler was implemented that balances the resource utilization rate, and the user's waiting time.

Overall, the designed approach will provide a platform to engage and coordinate many groups in the simulation community and advance the service in a more standardized and

reproducible pattern. Thus, this evolving community and toolset will narrow the gap between the service provider and the user.

7.2 Future Work

Future research may focus on: 1) a more refined description of a simulation service based on data input and parameters, 2) a feature extracting model to identify factors that significantly influence the simulation run time, and 3) development of a forecast model that considers key features to better estimate the task running time.

Reference

Albeck, J., & Gerken, J. (2017). *Traffic Signal Timing and Coordination Manual*. MNDOT.

Alvarez Mendoza, P. (2008). *Evaluating the effectiveness of signal timing optimization based on microscopic simulation evaluation*.

Andrews, J. B., & Lieberman, E. B. (1987). The NETSIM Graphics System. *TRANSPORTATION RESEARCH RECORD*, 8.

Beaulieu, M. (2007). *A GUIDE TO DOCUMENTING VISSIM-BASED MICROSCOPIC TRAFFIC SIMULATION MODELS* (WA-RD 678.1; p. 64).

Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). SUMO–simulation of urban mobility: An overview. *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*.

Bisong, E. (2019). Google colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 59–64). Springer.

Bocciarelli, P., D'ambrogio, A., Giglio, A., & Paglia, E. (2019). A microservice-based approach for fine-grained simulation in msaas platforms. *Simulation Series*, *2019-July*.

Boxill, S. A. (2007). *An evaluation of 3-D traffic simulation modeling capabilities* (SWUTC/07/167621-1). Texas Southern University. Center for Transportation Training and Research.

Buschiazzo, P., Leoncini, D., Zunino, R., & Scapolla, A. M. (2010). A web-based laboratory for digital signal processing. *International Journal of Online Engineering (IJOE)*, *6*(5), 6–11.

Byrne, J., Heavey, C., & Byrne, P. J. (2010). A review of Web-based simulation and supporting tools. *Simulation Modelling Practice and Theory*, *18*(3), 253–276. https://doi.org/10.1016/j.simpat.2009.09.013

Carini, R. N. (1977). *Application of the UTCS-1 Network Simulation Model to Select Optimal Signal Timings in a Multi-linear Street System: Interim Report*.

Chang, G.-L., & Kanaan, A. (1990). Variability assessment for TRAF-NETSIM. *Journal of Transportation*

Engineering, *116*(5), 636–657.

Chase, R. B. (1981). The Customer Contact Approach to Services: Theoretical Bases and Practical Extensions. *Operations Research*, *29*(4), 698–706. https://doi.org/10.1287/opre.29.4.698

Chen, K., Lu, W., Peng, Y., Rowlinson, S., & Huang, G. Q. (2015). Bridging BIM and building: From a literature review to an integrated conceptual framework. *International Journal of Project Management*, *33*(6), 1405–1416. https://doi.org/10.1016/j.ijproman.2015.03.006

Chung, K., Grembek, O., Lee, J., & Choi, K. (2013). Developing Safety Management Tools for State Departments of Transportation. *Transportation Research Record: Journal of the Transportation Research Board*, *2364*(1), 36–43. https://doi.org/10.3141/2364-05

Coronel, C., & Morris, S. (2017). *Database systems: Design, implementation, and management* (12e ed.). Cengage Learning.

Davies, G. W. (1972). *Optimization of a Traffic Signal System Through Computer Simulation*.

Dean, J., & Dean, R. (2021). *Introduction to programming with java: A problem solving approach* (Third edition). Mc-Graw-Hill Education.

Dowling, R. G. (2002). *California Department of Transportation: Guidelines for applying traffic microsimulation modeling software*. Dowling Associates.

Fellendorf, M., & Vortisch, P. (2010). Microscopic Traffic Flow Simulator VISSIM. In J. Barceló (Ed.), *Fundamentals of Traffic Simulation* (Vol. 145, pp. 63–93). Springer New York. https://doi.org/10.1007/978-1-4419-6142-6_2

FHWA. (n.d.). *FHWA Office of Operations—Appendix A: Introduction to CORSIM Theory*. Retrieved March 26, 2020, from https://ops.fhwa.dot.gov/trafficanalysistools/tat_vol4/app_a.htm

FHWA. (1996). *CORSIM User Manual Version 1.0.1*.

FHWA. (2004). *Transportation Management Plans (TMPs) for Work Zones—FHWA Work Zone*. https://ops.fhwa.dot.gov/wz/resources/tmp_factsheet.htm

FHWA. (2019). *Traffic Analysis Toolbox Volume III: Guidelines for Applying Traffic Microsimulation Modeling Software 2019 Update to the 2004 Version—FHWA Office of Operations* (FHWA-HOP-18-036). https://ops.fhwa.dot.gov/publications/fhwahop18036/index.htm

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Vol. 7). University of California, Irvine Irvine.

Forecasting, N. R. C. (US) C. for D. of the S. of the P. in M. A. T., & Board, T. R. (2007). *Metropolitan Travel Forecasting: Current Practice and Future Direction—Special Report 288* (Issue 288). Transportation Research Board.

Fundin, A., Witell, L., & Gebauer, H. (2012). Service transition: Finding the right position on the goods-to-services continuum. *International Journal of Modelling in Operations Management*, *2*(1), 69–88.

Gibson, D., & Ross, P. (1977). Simulation of Traffic in Street Networks. *Public Roads*, *41*(2), 80–90.

Goode, H. H., Pollmar, C. H., & Wright, J. B. (1956). *The use of a digital computer to model a signalized intersection*.

Halati, A., & Torres, J. (1990). Freeway Simulation Model Enhancement and Integration—FRESIM User's Guide. *Federal Highway Administration, Report No. DTFH61-85-C-00094*.

Hart, W. E. (2011). *Managing Scientific Workflows in Python with pyutilib.workflow*. 34.

Hranac, R., & Petty, K. (2007). Dashboards for Transportation Operations: Detector Health Case Study. *Transportation Research Record: Journal of the Transportation Research Board*, *1993*(1), 36–42. https://doi.org/10.3141/1993-06

Joan D. Sulzberg & Michael J. Demetsky. (1991). *Demonstration of TRAF-NETSIM for Traffic Operations Management* (FHWAlVA-92-R3).

Kallenberg, R., & Kallenberg, R. (2003). Managing the transition from products to services. *International Journal of Service Industry Management*, *14*(2), 160–172. https://doi.org/10.1108/09564230310474138

Katz, J. H. (1963). Simulation of a traffic network. *Communications of the ACM*, *6*(8), 480–486.

KeplerGL User guides. (2021). https://docs.kepler.gl/docs/user-guides

Kinney, J. R., & Ra, J. W. (1995). Training Entry-Level Engineers in Civil Engineering and Design Consulting Firm. *Journal of Management in Engineering*, *11*(3), 35–40. https://doi.org/10.1061/(ASCE)0742-597X(1995)11:3(35)

Kotler, P. (1980). *Marketing management: Analysis, planning and control / by Philip Kotler*.

Li, D., Mei, H., Shen, Y., Su, S., Zhang, W., Wang, J., Zu, M., & Chen, W. (2018). ECharts: A declarative framework for rapid construction of web-based visualization. *Visual Informatics*, *2*(2), 136–146. https://doi.org/10.1016/j.visinf.2018.04.011

Lieberman, E. (1991). *Integrating GIS, simulation and animation*. Institute of Electrical and Electronics Engineers (IEEE).

Lieberman, E. B. (2014). Brief history of traffic simulation. *Traffic and Transportation Simulation*, *17*.

Lieberman, E. B. (1970). Dynamic analysis of freeway corridor traffic. *MECHANICAL ENGINEERING*, *92*(12), 66-.

Lieberman, E. B., & Cohen, S. (1976). *New Technique for Evaluating Urban Traffic Energy Consumption and Emissions* (Issue HS-020 305).

Lu, W., Feng, W., & Huang, L. (2015). Integrated Simulation Platform of VISSIM, VC ++, and MATLAB. *ICTE 2015*, 1607–1613. https://doi.org/10.1061/9780784479384.202

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision Support Systems*, *51*(1), 176–189.

Mateljan, V., Čišić, D., & Ogrizović, D. (2010). Cloud database-as-a-service (DaaS)-ROI. *The 33rd International Convention MIPRO*, 1185–1188.

Mazen, I. (2013). *Business Models for Selling AEC Knowledge over the Cloud* [Thesis]. https://tspace.library.utoronto.ca/handle/1807/43205

McGhee, C. C., & Arnold, E. D. (1997). Review and Evaluation of Methods for Analyzing Capacity at Signalized Intersections. *Transportation Research Record: Journal of the Transportation Research Board*, *1572*(1), 160–166. https://doi.org/10.3141/1572-19

Mears, B. (1999). TSIS4.3 to be released this summer. *Center for Microcomputers in Transportation*, *15*, 6.

Miller, J. A., Fishwick, P. A., Taylor, S. J. E., Benjamin, P., & Szymanski, B. (2001). Research and commercial opportunities in Web-Based Simulation. *Simulation Practice and Theory*, *9*(1–2), 55–72. https://doi.org/10.1016/S0928-4869(01)00035-0

Nevada DOT. (2012). *CORSIM Modeling Guidelines*.

Olstam, J., & Tapani, A. (2011). A Review of Guidelines for Applying Traffic Simulation to Level-of-service Analysis. *Procedia - Social and Behavioral Sciences*, *16*, 771–780. https://doi.org/10.1016/j.sbspro.2011.04.496

Owen, L. E., Yunlong Zhang, Lei Rao, & McHale, G. (2000). Traffic flow simulation using CORSIM. *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, *2*, 1143–1147. https://doi.org/10.1109/WSC.2000.899077

Paksarsawan, S., Montgomery, F., & Clark, S. D. (1992). *How TRAF-NETSIM Works*.

pc-trans. (2003). *Pc-trans software price list*.

Pell, A., Meingast, A., & Schauer, O. (2017). Trends in Real-time Traffic Simulation. *Transportation Research Procedia*, *25*, 1477–1484. https://doi.org/10.1016/j.trpro.2017.05.175

Rajbhandari, R., Saman, S., Vadali, S., & Kang, D. (2012). Dashboard Tool to Communicate Delays and Economic Cost of Delays at International Border Crossings. *Transportation Research Record: Journal of the Transportation Research Board*, *2285*(1), 135–144. https://doi.org/10.3141/2285-16

Rathi, A. K., Santiago, A. J., & Valentine, D. E. (1989). *TRAF-EDIT: An interactive data editor for the TRAF simulation system*.

Remix. (2020). https://remix.com/

Robertson, D. I. (1969). *TRANSYT: A TRAFFIC NETWORK STUDY TOOL*. https://trid.trb.org/view/115048

Sampson, S. E. (2012). Visualizing Service Operations. *Journal of Service Research*, *15*(2), 182–198. https://doi.org/10.1177/1094670511435541

Santiago, A. J., & Kanaan, A. (1993). ATMS laboratories: A requirement for program delivery. *Surface Transportation: Mobility, Technology, and Society. Proceedings of the IVHS AMERICA 1993 Annual Meeting*.

Shang, S. S. C., Li, E. Y., Wu, Y.-L., & Hou, O. C. L. (2011). Understanding Web 2.0 service models: A knowledge-creating perspective. *Information & Management*, *48*(4–5), 178–184. https://doi.org/10.1016/j.im.2011.01.005

Shekhar, S., Abdel-Aziz, H., Walker, M., Caglar, F., Gokhale, A., & Koutsoukos, X. (2016). A simulation as a service cloud middleware. *Annals of Telecommunications*, *71*(3–4), 93–108.

Shi, X., Jin, J., Cheng, Y., Parker, S. T., Zhang, J., & Ran, B. (2013). Conceptual Design for a Research Oriented Web-based Traffic Simulation Platform. *92nd Annual Meeting of the 6 Transportation Research Board, Washington, DC*, *7*.

Sibley, S. W. (1985). NETSIM for Microcomputers. *Public Roads*, *49*(2), 54–59.

Silvestro, R., Fitzgerald, L., Johnston, R., & Voss, C. (1992). Towards a classification of service processes. *International Journal of Service Industry Management*, *3*(3), 0–0.

Sperling, D. (2015). *TRANSPORTATION RESEARCH BOARD 2015 EXECUTIVE COMMITTEE OFFICERS*. 156.

Stark, M. C. (1961). *Computer simulation of street traffic* (Vol. 119). US Department of Commerce, Office of Technical Services.

Tamminga, G., Miska, M., Santos, E., Van Lint, H., Nakasone, A., Prendinger, H., & Hoogendoorn, S. (2012). Design of Open Source Framework for Traffic and Travel Simulation. *Transportation Research Record: Journal of the Transportation Research Board*, *2291*(1), 44–52. https://doi.org/10.3141/2291-06

Towery, N. D., Machek, E., & Thomas, A. (2017). *Technology Readiness Level Guidebook*.

TransModeler. (2020). *TransModeler SE Free with HCS Streets highway capacity software*. https://www.caliper.com/transmodeler/transmodeler-se-hcs-traffic-intersection-analysis.htm

Transportation Networks for Research Core Team. (2020). *Transportation Networks for Research* [Jupyter Notebook]. https://github.com/bstabler/TransportationNetworks (Original work published 2016)

U.S. DEPARTMENT OF TRANSPORTATION. (2019). *Making Work Zones Work Better*. https://ops.fhwa.dot.gov/wz/about/wz_story.htm

U.S. Federal Highway Administration ; Texas A&M Transportation Institute. (2018a). *National Estimates of Total and Injury Work Zone Crashes*.

U.S. Federal Highway Administration ; Texas A&M Transportation Institute. (2018b). *Work Zone Fatal Crashes and Fatalities*. https://www.workzonesafety.org/crash-information/work-zone-fatal-crashes-fatalities/#national

Vandermerwe, S., & Rada, J. (1988). Servitization of business: Adding value by adding services. *European Management Journal*, *6*(4), 314–324.

Vaughan, C., Aghdashi, P. B., Carnes, C., Oyler, G., & Samet, R. (2017). *Video Collection of Traffic Violations for Public Education and Regulatory Changes*.

Warning messages and fatal error CORSIM. (n.d.).

Wicks, D. A., & Lieberman, E. B. (1980). *Development and Testing of INTRAS, a Microscopic Freeway Simulation Model, Volume 1: Program Design, Parameter Calibration and Freeway Dynamics Component Development*.

Wiedemann, T. (2001). Simulation application service providing (SIM-ASP). *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*. https://doi.org/10.1109/WSC.2001.977347

Xie, G., & Hoeft, B. (2012). Freeway and Arterial System of Transportation Dashboard: Web-Based Freeway and Arterial Performance Measurement System. *Transportation Research Record: Journal of the Transportation Research Board*, *2271*(1), 45–56. https://doi.org/10.3141/2271-06

Yang, Q., Slavin, H., Stefansson, K., Rabinowicz, A., Olsberg, S., LaClair, M., & Brandon, J. (2006). *Traffic data management and simulation system*.

Zhang, J. (2011). *A Social Semantic Web System for Coordinating Communication in the Architecture, Engineering & Construction Industry* [Thesis]. https://tspace.library.utoronto.ca/handle/1807/26486

Zhang, L. (2016). *Microscopic Traffic Simulation Models and Software: An Open Source Approach, Tech Brief*. Federal Highway Administration.

Zhang, L., Ghaman, R., & Morales, J. M. (2002). Application of Distributed Traffic Simulation to Evaluate Traffic Signal Controller Algorithms. *Traffic And Transportation Studies (2002)*, 980–987. https://doi.org/10.1061/40630(255)137

Zhang, L., Morales, J. M., & Ghaman, R. (2002). An Application of Client/Server Architecture in Interfacing Virtual Traffic Signal Controller with CORSIM. *Applications of Advanced Technologies in Transportation (2002)*, 836–843. https://doi.org/10.1061/40632(245)105

Zhou, B., Blum, J., & Eskandarian, A. (2005). Virtual reality visualization of microscopic traffic simulations. *Transportation Research Record*, *1937*(1), 159–166.

Zhou, X. (2021). *Asu-trans-ai-lab/Integrated_modeling_GMNS* [Jupyter Notebook]. https://github.com/asu-trans-ai-lab/Integrated_modeling_GMNS (Original work published 2021)

Zhou, X., & Jiangtao, L. (2015). *Data Structure and Workflow of DTALite_NeXTA.pdf*.

Zhou, X., & Nevers, B. (2014). *Introduction to AMS Data Hub.pdf*.

CURRICULUM VITAE

Zihao Jin

EDUCATION

Ph.D., University of Wisconsin-Milwaukee	Aug. 2021
B.S., Shanghai University of Engineering Science	June 2016

PUBLICATION

Yuan, Yun, Jie Yu, Dahai Han, Weijie Tan, and **Zihao Jin**. 2018. "Strategic Planning of Urban Network Evacuation with Balanced Signal Control and Crosselimination Strategies." Journal of Transportation Engineering, Part A: Systems 145 (1): 04018080.

Han, Dahai, Jie Yu, Edward Beimborn, **Zihao Jin**, and Weijie Tan. 2019. "Elements of Successful Universal Student Transit Pass Programs from Planning to Implementation: A Benchmark Study." Transportation Research Record: Journal of the Transportation Research Board 2673 (4): 833–43.

Shamsi Trisha, Jie Yu, **Zihao Jin**, X. L. 2020. "Smart Mobility, Smart Campus: A Framework for Integrating Transportation Services at a Major Trip Generators." 99th TRB Annual Meeting Proceedings.