DYNAMIC COMPOSITION AND MANAGEMENT OF EMERGENCY RESPONSE PROCESSES

By

ABEER ELAHRAF

A Dissertation submitted to the

Graduate School-Newark

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Management

written under the direction of

Dr. Jaideep Vaidya & Dr. Basit Shafiq

and approved by

_____

_____

_____

_____

Newark, New Jersey, USA

October 2021

Copyright page:

ABSTRACT OF THE DISSERTATION

Dynamic Composition and Management of Emergency Response Processes

By ABEER ELAHRAF

Dissertation Directors:

Dr. Jaideep Vaidya & Dr. Basit Shafiq

The number of disasters, whether human-made or natural, has been on the rise over the recent years; they have destroyed lives, homes, and caused widespread damage to infrastructure. The increase in disasters means there is also a higher demand for efficient response planning. Effective emergency response planning requires communication and coordination with the diverse operational systems belonging to various collaborating government agencies, non-government organizations (NGOs), and private sector entities. An essential requirement for developing an emergency response process is to establish information sharing and system-level interoperability among the operational systems of collaborating organizations. The challenge is that emergency response processes are not well-structured and do not have a well-defined outcome; they are knowledge-centric. Their workflow structure and execution may evolve dynamically based on the environmental context and the type of service or activity invoked during process execution. It is impractical to define static plans and response process workflows for every possible situation since unforeseeable situations may arise. Thus, a dynamic response requires adaptability to a changing situation as an

incident evolves.

In this thesis, we develop an iterative end-to-end solution for the dynamic composition and management of an Emergency Response Management System, called Dynamic Emergency Response Processes System (DERPS). DERPS allows the Incident Commander to develop a contextualized response using ontology-based reasoning and allows its dynamic adaptation to situational changes. We also adapt and apply DERPS in the COVID-19 context. Specifically, given the pandemic's scale and scope, home-based isolation has been considered a potential first step to reduce the spread of the virus and limit the stress on the healthcare system. However, home-based isolation needs and requirements are unique for each patient. We show how DERPS can be utilized to develop personalized patient care plans to ensure that each patient's needs are appropriately met while they are confined to their home. We develop a prototype implementation to show the feasibility of the proposed framework and discuss challenges and issues in deploying such a system in practice.

Acknowledgments

In the name of God, the merciful and compassionate. I would like to praise and thank Allah for guiding me through my life, especially during my Ph.D. program!

Secondly, I want to express my deepest gratitude and appreciation to my doctoral advisors, Dr. Jaideep Vadiya and Dr. Basit Shafiq. My appreciation is also extended to the committee members, Dr. Vijayalakshmi Atluri and Dr. Shamik Sura. Your insightful advice and guidance helped me finish this dissertation. I particularly enjoyed my challenging conversations with Dr. Vadiya and Dr. Shafiq; these conversations, were filled with novel curiosities and encouragements. I also would like to thank Dr. Nabil Adam for his help and guidance, especially during my early years in the Ph.D. program.

I would like to thank my co-workers, fellow faculty members, and colleagues at Rutgers University, particularly the current and past staff of the SASN Dean's office, for their support and encouragement.

Last, but not least, to my steadfast husband and companion, Amr Farag, I sincerely thank you for your unconditional love, support, encouragement, guidance, and patience. I could not have achieved this remarkable result without your faith in me and continues dedicated support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Objective

An emergency response process is a workflow of activities that need to be executed in response to a critical situation [44]. Effective emergency response planning requires communication and coordination with the various operational systems of different collaborating organizations, including government agencies, non-government organizations (NGOs), and private sector entities. Establishing information sharing and system-level interoperability among organizations' various operational systems is an essential requirement for developing an emergency response process. Unlike the traditional business processes (e.g., e-government and e-commerce processes), which have well-defined and predictable workflow structure and outcomes, emergency response processes are knowledge and data-centric and are executed in a dynamic and non-deterministic environment. Their workflow structure emerges and evolves dynamically, depending upon the environmental context, user decisions, availability of resources, and the jurisdictions' and response organizations' rules and policies. Establishing interoperability is particularly challenging in such an environment; wherein, the workflow structure, collaborating organizations, data sources, and resource providers may not be known *a priori* and would need to be identified at runtime. Moreover, there may be new organizations that have not collaborated earlier. Therefore,

an approach to enable the on-the-fly composition of emergency response processes by enabling information sharing and interoperability among the information systems of relevant response organizations is needed.

In a disaster situation, it is essential to have an emergency response plan that defines the framework of necessary activities. Each incident is unique in its location, circumstances, environmental condition, sequence of events, and the response organizations' jurisdictions and policies, which makes it impractical, if not impossible, to predefine emergency response plans for every situation. Substantial research effort exists on the topic of emergency response management systems [18, 23, 38, 45, 53, 54] as well as the workflow management systems [44, 52] to improve the response plans and responsiveness to emergency incidents. Previous research has assumed prior knowledge of participating organizations/agencies and pre-existing system-level interoperability between them. Research topics have focused on resource scheduling and the critical risk of the performance analysis in an emergency rather than the risks posed to human lives and natural resources. It is essential to customize the response plan based on the disaster characteristics, location, and organizations/agencies with the resources to perform the necessary tasks.

Consider the following scenario that illustrates the need for the proposed system. A freight train colliding with a car at a cross-track in Saddle Brook, New Jersey is causing a sizeable greenish smoke cloud to emanate from the wreckage. This incident requires fire trucks, firefighters, ambulances from local and most likely neighboring towns with different jurisdictions and policies. In addition, the train's contents are unknown, which limits understanding of the severity of the situation. Our approach would dynamically generate an executable response plan based on available resources in Saddle Brook using the Web services for: 1) Saddle Brook Fire Dept (in house APIs), fire trucks, firefighters, and ambulances; 2) askRail for train content; 3) CAMEO for chemical information; and 4) ALOHA for Hazard Modeling.

Towards this goal, we developed an integrated approach that enables dynamic composition of response processes by interfacing the operational systems of different response

organizations for response process composition and management. Our proposed solution, called Dynamic Emergency Response process System (DERPS), provides the ability to compose, refine, deploy and execute an emergency response plan iteratively until the disaster is resolved. This solution is built on a service-oriented architecture and requires that response organizations provide Web service APIs to access their information and resource management systems.

1.2 Environment

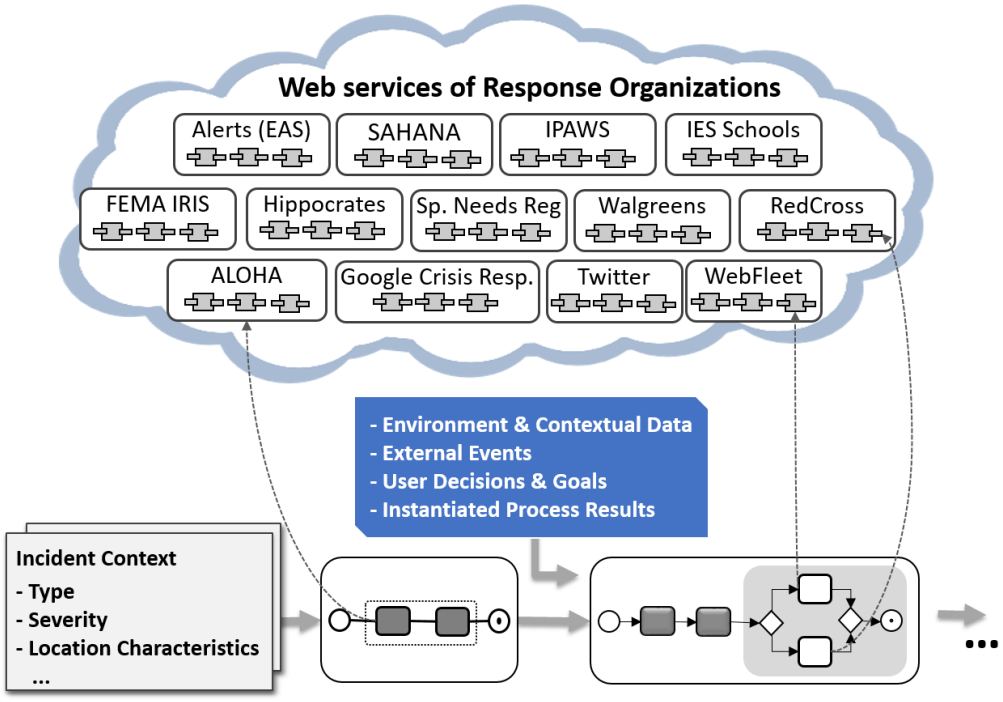

Figure 1.1: Environment for Emergency Response Process Management

Today's service-oriented technology has enabled organizations to utilize the cloud infrastructure to provide their customers with various online services. With the assumption that all services are easily accessible by public Web services, let us review our proposed environment in Figure 1.1. This figure exhibits the services cloud environment for dynamic composition, execution, and management of emergency response processes. The response process is composed by integrating the diverse systems and APIs of the organiza-

tions participating in the response activities. These organizations may use systems that are developed in-house, proprietary, or cloud-based with various APIs to perform the systems' underlying functional activities. Furthermore, as the situation evolves, new requirements need to be incorporated to dynamically compose an extended response process with new activities. These changes require evaluating the latest status, identifying the agencies and organizations that can provide the required resources, and integrating their Web services into the instantiated response process.

1.3 Challenges and Problem Statement

Our research focuses on the dynamic composition of emergency response processes, which involves information sharing and interoperability among operational systems of the government and non-government organizations.

Establishing System-Level Interoperability Across Organizations: In an emergency, it is essential to promptly respond to the incident and swiftly restore normality, making time a crucial element in the response solution. The emergency response process involves information sharing and interoperability among operational systems of governmental and non-governmental organizations. There are many agencies/organizations involved that will be using different systems such as components developed in-house, pre-compiled libraries, or open-source platforms. Communication among these system-level components must be resolved and established to be effective for this plan. We resolve this problem using the organization's Web services. We utilize our prior work [3] to resolve heterogeneity between the input/output parameters of services to integrate the APIs of various organizations, and addressing this challenge in our work resolves system-level interoperability.

Dynamic Composition of Processes Workflow Structure: The collaborating organizations, data sources, and resource providers may not be known *a priori* and may need to be discovered at runtime. The challenge of the composition lies in the available Web services/APIs since they are discovered dynamically based on the incident and the responder(s) available resources. Therefore, our approach first determines the required activities

for the incident based on its characteristics; these activities need to be performed through organizations' available resources. Secondly, we discover and select the relevant APIs of the organizations' operational systems to perform the activities. Lastly, we facilitate an on-the-fly composition of emergency response processes workflow by considering the dependencies between activities and identifying Web services' invocation sequence. We address this challenge and resolve it in our work.

Process Adaptation and Evolution: Incident environmental context due to the incident evolution or user decision, and agency/organization resource availabilities may change during the incident. A successful response process plan needs to dynamically change and accommodate these continuous and various changes, which produce a challenge of generating a stateful approach for developing an executable response process that adapts and evolves dynamically. As new events arise, our approach determines additional activities for the incident and creates a new dynamic composition of processes iteratively. Also, if resources are available at the local-level (city/town) agencies, the search expands to the county and state-levels agencies. This dynamic re-composition will continue to execute until the incident is resolved. Our approach has resolved this challenge as it can adapt to evolve dynamically to the incident evaluations.

In this thesis, We also extend this framework to a COVID-19 scenario, where a patient has pre-existing conditions. Here, the framework is successfully applied as a prototype. However, for a successful deployment of such a system to a real-world environment, additional challenges need to be considered, including scalability, privacy and security, interoperability and standardization, and auditability. We do not address those additional challenges.

1.4 Research Contributions

The key research contributions of this dissertation can be summarized as follows:

1. We propose an integrated approach that facilitates the dynamic composition of exe-

cutable response processes. Our approach develops an iterative end-to-end solution for the dynamic composition and management of an Emergency Response Management System.

- We have developed the DERPS that supports dynamic composition, adaptation, and evolution of response processes by considering the incident characteristics, operational environment, occurrence of any new events, availability of response resources, and user decisions.
- We developed a prototype implementation of the proposed framework's functionality for dynamic composition and management of emergency response processes.
- We performed an extensive experimental evaluation of the proposed system and its effectiveness using a rail incident scenario similar to the Federal Emergency Management Agency's (FEMA) Hazardous Materials Tabletop Exercises Manual Rail Incident Scenario [22].

2. We evaluated the feasibility of our proposed approach to generate personalized patient-care plans for COVID-19 patients under home-based isolation. Home-based isolation has been considered a potential first step for reducing both the spread of the virus and the healthcare system's stress. Therefore, it is necessary to develop personalized patient care plans to ensure that each patient's needs are appropriately met. We have applied our approach as a service-oriented framework that allows dynamic composition and management of such patient care plans assuming the existence of an appropriate knowledge base and availability of Web services interfaces of the underlying systems of caregivers and service providers.

   - We developed a prototype implementation to show the feasibility of the proposed framework for continuous monitoring of the care plan. This monitoring allows for a reassessment of the patient's evolving needs, preferences, and treatment responses to interventions and re-composition of the care plan.

1.5 Outline of Dissertation

The dissertation is organized as follows. In Chapter 2, we review the related work on Emergency Response Management Systems and Frameworks, Schema Evolution Requirements, Runtime Process Adaptation Requirements and Knowledge-Driven Business Process Development. In Chapter 3, we discuss our proposed Dynamic Composition and Management of Emergency Response Processes. In Chapter 4, we discuss the application of our proposed approach to develop personalized Patient Care Plans for COVID-19 patients. Chapter 5 presents our prototype implementation and experimental evaluation for our two proposed systems. Finally, in Chapter 6, we provide the conclusion and discuss future work.

Chapter 2

Related Work

In this chapter, we discuss the related work on automated service composition approaches in the context of emergency response processes. We first review the existing emergency response management systems and frameworks. Next we discuss the key requirements and related approaches for dynamic process composition.

2.1 Emergency Response Management Systems and Frameworks

There are some service-oriented platforms and systems for interoperable information exchange for emergency management and response planning including, XchangeCore[1], formerly known as Unified Incident Command Decision Support System (UICDS) [38, 45], FEMA Incident Resource Inventory System (IRIS) [23], Service-Oriented ArchiteCtures Supporting Networks of Public Security (SoKNOS) [18], and social media alert and response to threats to citizens (Smart-C) [2]. Below we briefly discuss each of these platforms/systems and then provide a comparison of these platforms/systems.

UICDS framework for Incident Management/XchangeCore: The Department of Homeland Security Initiative developed the UICDS framework for information sharing also known

[1] https://www.saberspace.org/xchangecore-home.html

as XchangeCore. The framework's goal is to standardize the information-sharing infrastructure in minor and significant emergencies by enabling a user's device to get answers, push alerts, ask questions, and other functions. XchangeCore is designed to work on both small and wide-area networks. It also includes risk decision support, which is done via geospatial predictive analytics to detect and suggest a high-risk area for crimes or disasters. Shafiq et al. used the UICDS framework in their research [46] to create a prototype for an incident management middleware across various organizations (government and non-government). The goal of this framework is to provide the emergency responders with a plan, critical information, essential incident-related information, and to store incident information for later analysis. The solution is an ontology-based resource that consists of: 1) an ontology library with the incident types and resources; and 2) a reasoning engine to evaluate the required resources based on the incident's context information and policies.

Social Media Alert and Response to Threats to Citizens (SMART-C): In this research Adam et al. proposed a plug-and-play system capable of collecting data from various sources to analyze it and produce an easily comprehensible of the incident to systems and different users based on relevant rules and policies. The SMART-C system is a part of the social media initiative at the Department of Homeland Security.

Table 2.1: Comparison of Interoperable Systems for Emergency Management and Response Planning

	Information sharing	Reasoning support	Dynamic discovery of response organizations	Automated process composition
UICDS/XchangeCore	Yes	No	No	No
IRIS	Yes	Yes	No	No
SoKNOS	Yes	Yes	Yes	No
Smart-C	Yes	Yes	Yes	No
Proposed Approach [20]	Yes	Yes	Yes	Yes

The Incident Resource Inventory System (IRIS): IRIS is an inventory system provided by FEMA. It is available to all government and non-government organizations/agencies. A consistent resource inventory database tool, IRIS allows access to any resources for incident operations and mutual aid purposes.

SoKNOS - An Interactive Visual EM Framework: Doweling et al. suggest an emergency management framework, SoKNOS, a project funded by the German Federal Ministry of Education and Research. For this framework, they introduce an approach to develop a system that would allow users to create a structural and temporal view of the integrated emergency information data based on their role. The system would have various source inputs and integrate them into repositories, such as information about the incident, the weather, geospatial services, organizations involved, Web services, and internally-created data. The users would interact with the system via plugins for temporal or structural views. To resolve the interoperability between organizations, they assume that the Web services' underlying information is described using ontology.

Table 2.1 compares these platforms/systems and their capabilities to support the dynamic composition of emergency response processes. Specifically, we compared the capabilities of these platforms/systems in terms of: 1) information sharing among diverse systems; 2) reasoning support for response planning; 3) dynamic discovery of response organizations; and 4) automated process composition by interfacing with the various operational systems of response organizations. Although these systems employ semantic-based reasoning for emergency response planning and decision support, they do not support the on-the-fly composition of response processes that may evolve based on the situation at hand.

2.2 Dynamic Process Composition

Automated business process composition and management have been extensively studied in the literature. Significant research has been carried out on dynamic service composition,

process adaptation, and schema evolution in the context of structured business processes. However, there is a lack of integrated approaches that address the dynamic composition requirements for knowledge-driven processes that are unpredictable and emergent. Below we discuss the essential requirements and related approaches for dynamic process composition. We discuss the work that tackles the issues of schema evolution, runtime process adaptation, and knowledge-driven business process development for completeness. However, our work primarily tackles the issues in knowledge-driven business process development. We do not handle schema evolution or service failure/replacement issues since these can be be addressed using prior research.

2.2.1 Schema Evolution Requirements

Schema evolution requirements mostly arise due to process re-engineering efforts such as business rules, policies, regulation, business strategy, and continuous process improvement. Such changes in schema typically require developing a new version of the process schema/model, which, in turn, affects the running instances of the process as well [48]. The Schema change problem has been extensively studied, and several approaches have been proposed. These include graph-based model matching and merging techniques [6, 11, 31, 51], as well as rule-based techniques [21, 36, 41, 55]. Graph-based techniques support flexibility in process models to enable process designers to compare and reuse configurable elements [6] in existing process models. These approaches mainly address design time variability in process workflows. On the other hand, rule-based approaches primarily support dynamic changes in process schema by enabling different ad-hoc deviations from a given process model. Table 2.2 compares these platforms/systems concerning their capabilities to support schema evolution and runtime process adaptation systems for emergency management and response planning.

Bucchiarone et al. have proposed a series of works on *Context Evolution Based Process Adaptation* capable of handling stateful and non-deterministic services. Their work proposed a goal-oriented business process approach that verifies the evolution of the pro-

cess context and execution against the desired goal-based model that encodes the business policies over the domain elements [10]. In the case of divergence of the desired goal model, the adaptation process is triggered. An automatic composition and execution are generated considering the available services at the current state and context. Bucchiarone et al. proposed another framework for context-aware dynamic composition of process fragments in their later work [8, 9]. In this work, the composition requirements are specified on the context properties, making this approach very useful in a dynamic environment where the available process fragments [6] and the execution context may change. The underlying idea is that different entities (service providers) in the system publish their functions through process fragments that can be used in new process compositions and dynamic process evolution.

2.2.2 Runtime Process Adaptation Requirements

Runtime process adaptation requirements primarily arise due to issues such as: 1) abnormal termination or failure of tasks (e.g., underlying Web service failure or unavailability); 2) violation of any constraints related to data (e.g., missing input) or tasks (e.g., pre/post-condition not satisfied); and 3) temporal requirements (e.g., expiration of a deadline) [48]. These requirements have been addressed by flexible and adaptive service composition approaches that mainly rely on exception handling mechanisms with the associated recovery procedures built manually by a process designer at runtime.

Exception Handling Processing Adaptation: The works of Reichert et al. [41] and Redding et al. [40], and their approaches to process adaptation, rely on exception handling mechanisms paired with recovery procedures that are built manually at runtime by a process designer. The adaptive approach used by Reichert and his team in their system ADEPT2 supports handling unanticipated exceptions by enabling different ad-hoc deviations from a given process model. The system identifies structural conflicts that would indicate a change in the schema. It then adapts the indicated change to the design of the business processes. Redding et al. had a different approach; they did not structure the business objects as struc-

Table 2.2: Comparison of Schema Evolution and Runtime Process Adaptation Systems for Emergency Management and Response Planning

	Manual	Pre-Planned	Unplanned	Non-deterministic Stateful Service
Bucchiarone et al. (UMS) [8, 9, 10]	-	Yes, Goal-based model	Yes, Process fragments dynamic composition	Yes
Marella et al. (SmartPM) [36]	Yes, by Expert on process domain	Yes, pre-planned exceptions	Yes, Situation Calculus logic framework & Automated Planning Techniques	Yes
Redding et al. [40]	-	Yes, Object-centric meta-model	-	-
Reichert et al. (ADEPT2) [41]	-	Yes, Case-based reasoning (CBR)	-	-
Yu et al. (MoDAR) [55]	-	Yes, Model-driven using aspects & rules	-	-

tural "flowchart-like notation" but rather as "what can happen during a case." Although their approach is still based on exception handling, it is not the typical structure.

The SmartPM Framework, an adaptation of business processes developed on the Knowledge Representation and Reasoning (KR&R) technique, was proposed by Marella et al. [36]. The framework adapts to errors (exception handling) during execution using the expected reality and actual reality models. The framework uses situation calculus, logic-based programming, and planning to recover from the potential gap between the two reality

models. The framework is developed to support monitoring and adaptation of controlled domains (physical and expected reality), but it does not support automated process adaptation. Its current implementation relies on a process designer for the specification of the process participants and model. However, an automated composition approach can be integrated with SmartPM to replace the Business Process Model and Notation (BPMN) based manual process design approach.

Adaptive Service-Based System Using Aspects & Rules: Yu et al. introduced a model-driven approach, MoDAR, based on aspects and rules to build an adaptive service-based system [55]. The approach separates business processes into two parts: 1) the variable part, defined as business rules; and 2) the stable part, defined as the base part. Lastly, rules are weaved into the base process. This approach is considered adaptive because the business rules (variable part) can change at run time, and the base part (stable part) will not be affected.

In this thesis, we consider process adaptation only in the sense of dynamic recomposition of emergency response processes to respond to changes in the environment or the emergence of new requirements. We do not focus on service replacement/failure and schema changes and assume that the existing approaches discussed above can be employed to handle such issues. However, we also utilize a rule-based adaptive process composition approach to address such issues from an implementation perspective.

2.2.3 Knowledge-Driven Business Process Development

Emergency Response Management Systems, Decision Support Systems, and Medical Diagnosis Systems are examples of knowledge-driven and data-centric decision systems. They contain a limited structure since scenarios might already have pre-existing plans and activities. Still, an exact order of execution will only be determined at the time of performance based on the "knowledge" and the "data" provided. These types of systems involve taking into consideration three main characteristics of knowledge-driven processes which differentiate them from structured business processes: 1) Unpredictable and emer-

gent process workflow structure: the activities and process structure are not predefined and are determined dynamically based on the knowledge of environmental context, situation, and case-specific parameters that are not known *a priori* and are subject to change at runtime; 2) Goal-oriented composition: composition goals may change, and new goals may arise as new data or actions emerge during process instantiation and execution; and 3) Constraint and rule-driven composition: policies, rules, and regulations can influence the process structure and constrain its execution. While there is a lack of integrated approaches addressing these requirements, some works address different needs in knowledge-driven business process development in a piecemeal manner [6, 8, 9, 10, 32, 36]. Table 2.3 compares these platforms/systems concerning their capabilities to support knowledge-driven BP composition and management systems for emergency management and response planning.

Below we review some knowledge-driven BP composition and management systems:

Mobile-Based Emergency Response System (MERS): MERS, a decision support system implemented by Amailef et al. [5] using an ontology-supported case-based reasoning approach. The system has four core components: 1) data acquisition, which includes mobile data entries(SMS); 2) ontology, which is used for structure; 3) knowledge base, which contains all the past incidents; and 4) reasoning, which compares the current incident with the knowledge and solutions from past incident events and provides a result of similar incidents with similarity scores (0-1). MERS reasoning component allows for *adaptation*, which enables the user to revise the solution until it is suitable for the current incident. Lastly, MERS also allows solutions to be saved and tagged with a solution rating (acceptability, flexibility, completeness, adequacy, feasibility) for reusability, updating the case-base for future query searches.

Decision-Making Model Utilizing HTN Approach: Tang et al. [49] proposed a solution for coordinating emergency response plans among multiple organizations during an emergency by integrating Hierarchical Task Network (HTN) planning and scheduling technolo-

Table 2.3: Comparison of Knowledge-Driven BP Composition and Management Systems for Emergency Management and Response Planning

	System Type	Modeling/ framework	Systems predefined	Repository
Amailef et al. (MERS) [5]	Mobile-based	Ontology-supported Case-based reasoning (OS-CBR)	Yes	No
De Nicola et al. [16]	Scenario Modeling	Ontology, Semantic-based,	Yes	No
Moreira et al. [37]	Disaster Management	ontology-driven situation-aware (SA) disaster management (DM) framework	-	No
Tang et al. (HTN) [49]	Scenario Modeling	HTN AI planning; Ontology, semantic characterization & pattern-based approach	Yes	Yes

gies. HTN [25] is an artificial intelligence (AI) planning technique that provides a mechanism that easily converts domain knowledge at different levels and manipulates it to refine non-primitive tasks into smaller subtasks until they are all executable. Tang et al.'s research extended this approach by developing four planning requirements: 1) hierarchical task structure vital for multiple responding organizations to execute tasks/subtasks; 2) plan validity, which is essential and may occur due to highly dynamic environment changes; 3) interdependencies and synchronization between planned activities, which are crucial in case of a cause-effect relationship between activities; and 4) temporal constraints for reasons such as deadlines, partial order relationships, and unanticipated events (i.e., weather, traffic). This technique provides valid results, but assumes that the organizations/agencies are already known.

An EM Scenarios Application for Smart Cities: De Nicola et al. proposed a scenario modeling software application to query, generate, and organize sets of conceptual models that then can be adapted to fit the incident in the smart city scenario [16]. The application has three main goals: 1) to provide an understanding of the consequences of decisions on potential crises; 2) to provide a training tool for crisis operators and analyzing EM plans for experts; and 3) to provide a foundational definition of executable simulation models of EM scenarios. This framework is built on three types of knowledge: 1) structural knowledge, which can be considered the scenarios' blueprints; 2) domain knowledge, which refers to the smart city and emergency management; and 3) contextual knowledge, which can be geographical, laws, or aspects of regulations that are needed for reasoning purposes (e.g., metro service does not operate on holidays).

Ontology-Driven Situation-Aware Disaster Management: Moreira et al. discuss their development of a new ontology-driven situation-aware (SA) disaster management (DM) framework [37]. Their work concentrates on the importance of a well-founded DM core ontology and the challenges of arranging concepts related to modeling situations in a foundational ontology. The work also describes how the foundational ontology can offer proper modeling languages and their guidelines to assist SA applications' development. Their framework is focused on the situational concept of SA application at design time and runtime. Moreira et al. argue that the core ontology continues to bring challenges for the Unified Foundational Ontology (UFO) and its ontological language (OntoUML) and purpose as it adapts to a mature foundational ontology with high-level concepts in related situations. They also urge the need for a theoretical foundation for modeling language constructs to support the SA application's specifications.

2.3 Conclusion

Most of the existing composition and management systems for emergency management and response planning are not highly dynamic. They cannot handle continuous non-deterministic

stateful service incidents, or incidents with a high frequency of unexpected developments. Most of these systems cannot automatically adapt and evolve without relying on pre-existing exception handling or the intervention of domain experts.

Chapter 3

Dynamic Composition and Management of Emergency Response Processes

In this chapter, we discuss the proposed approach for a Dynamic Emergency Response Processes System (DERPS), which supports dynamic composition, adaptation, and evolution of response processes. DERPS provides this support through consideration of the incident characteristics, the operational environment, occurrence of any new events, availability of response resources, and user decisions. The proposed approach applies ontology-based reasoning to determine the default actions and resource requirements for any given incident. It identifies relevant response organizations and APIs of their operational systems based on their jurisdictional and mutual aid agreement rules.

3.1 Introduction

As discussed in Chapter 1, emergency response processes are knowledge-driven and are executed in a dynamic and non-deterministic environment. Their workflow structure emerges and evolves dynamically depending upon the environmental context, user decisions, availability of resources, and the rules and policies of response organizations. These response organizations may not have pre-established collaboration and are discovered at runtime. Therefore, there is a need to facilitate on-the-fly composition of emergency response pro-

cesses by enabling information sharing and interoperability among the information systems of relevant response organizations. This is also illustrated in the following scenario.

Consider a rail incident scenario similar to FEMA's Hazardous Materials Tabletop Exercises Manual Rail Incident Scenario [22], in which a freight train is reported to collide with a car at a crossing. First responders approach the area to find a large greenish smoke cloud emanating from the wreckage. An initial incident command structure is established, and the incident commander (IC) determines the following response actions:

- Request dispatch of initial response resources locally from the city (including EMS, law enforcement, fire, and HazMat teams).
- Determine the risks posed by the hazards and identify threat zones and safe distances.
- Order evacuation of the hazard area as per the identified threat zones.
- Notify appropriate regulatory agencies, schools, and the general public within the area through Emergency Alert System (EAS) of the evacuation and shelter information.
- Activate other public warning systems (e.g., social media and radio alerts).
- Identify local resources that can be committed and are ready to dispatch.

The situation further evolves when the city Dispatch Center informs the IC of the expected arrival of a northbound passenger train in half an hour. Responders assisting with the evacuation also report that numerous people need specialized transportation (i.e., wheelchair-bound, hearing and vision impaired, on oxygen, bedridden). Based on the current situation, the IC decides to take the following actions:

- Evaluate the need for additional resources to deal with traffic control (i.e., portable road signs, redirect traffic flow patterns, and limit access).
- Instruct the railroad company to take steps for halting all incoming train traffic.

- Request resources to assist with the special needs evacuation.
- Request additional resources from other states to assist with the evacuation (i.e., transportation, food, and medical supplies).

Similarly, the IC may take additional actions as the situation evolves. These actions correspond to extending the instantiated response process with more activities.

Currently, there is no unified system to support automatic interfacing with the operational systems of response organizations for response process composition and management. Moreover, existing approaches for adaptive process composition [43, 48, 55] are limited to structured BPs, which employ event-condition-action rule-based reasoning for process adaptation. The rules are defined for different types of events (e.g., replacing a failed service by a pre-defined service). However, emergency response process management involves certain types of events for which the concrete actions may not be known *a priori*. For example, if a given resource request cannot be satisfied by a local agency, then the request is sent to a state-level agency. However, the specific state-level agencies and their service APIs need to be determined at runtime. Such events trigger recomposition of the emergency response process, which is the main focus of this paper. Our key contribution is to ensure that the needed process of recomposition can be automatically and dynamically performed as the situation evolves.

Our proposed approach provides an integrated framework for dynamic composition of an executable response process. It employs ontology-based reasoning to determine the default actions and resource requirements for the given incident and to identify relevant response organizations and APIs of their operational systems based on their jurisdictional and mutual aid agreement rules. The discovered Web service APIs of the different response organizations are then used to generate an executable response process that evolves dynamically based on any changes in the environmental context as well as the availability of resources. We experimentally validate the effectiveness of the proposed approach using an example scenario derived from FEMA's Hazardous Materials Tabletop Exercises

Manual.

In the following sections, we discuss each component of the proposed system in detail.

3.2 Problem Statement

We can formalize this problem as follows:

Given:

i. *the contextual information of the incident including incident type, severity, and location;*

ii. *the default actions or static response plans pre-defined for specific emergency events;*

iii. *response organizations, their jurisdictional and mutual aid agreement rules, and the resources they can provide;*

iv. *available Web services/ APIs of the operational systems of response organizations to request their resources;*

compose an executable response process that can be instantiated for the incident at-hand, and that can adapt and evolve as the incident unfolds or as the environmental context changes.

An executable response process refers to a business process in which each activity is bound to the appropriate operational system Web service/API and is deployed on a process engine for instantiation. We assume that the functionality of the underlying operational systems of response agencies and resource provider organizations is exposed through their respective Web service operations. Therefore, each activity in the abstract workflows of default actions is realized through a composition of Web service operations of the potential response organizations.

3.3 Proposed Approach

The proposed approach for dynamic composition and adaptation/evolution of an emergency response process is a multi-step approach that incrementally generates an executable response process and enables adaptability to a changing situation as the incident evolves. Figure 3.1 depicts the architectural components of our proposed approach. A vital component of the proposed system is the ontology with reasoning support. The ontology characterizes incidents based on 1) type, severity and location; 2) the default actions and resources required for resolving different incident types; and 3) the response agencies and organizations characterized based on their jurisdictions, roles and responsibilities, and the type of resources they can provide.

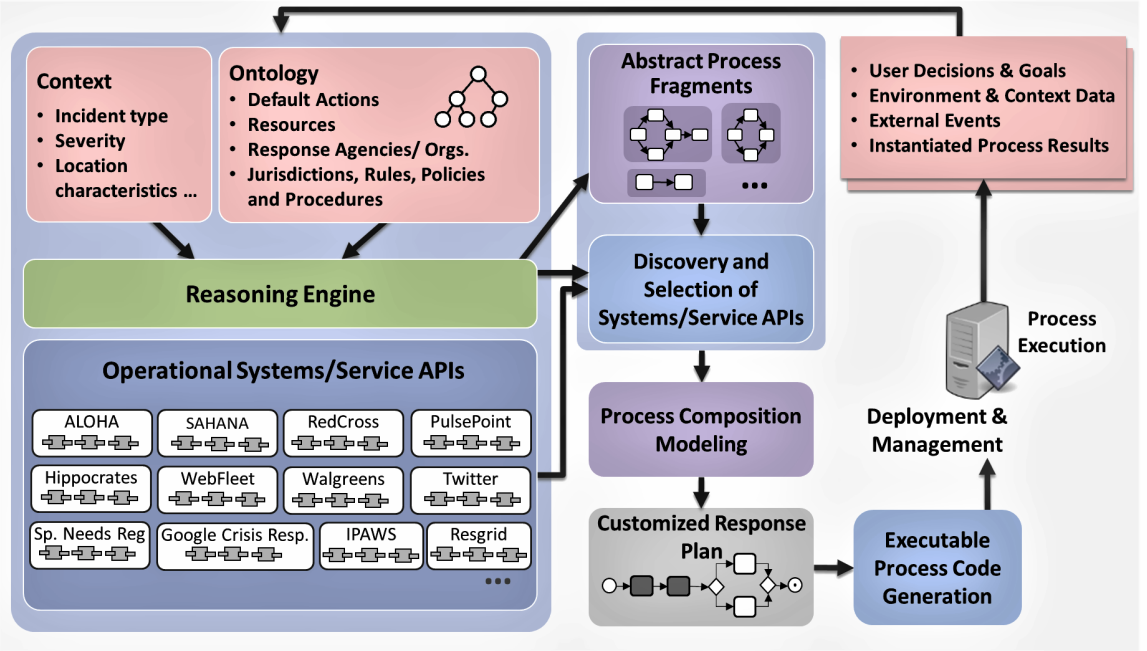

Figure 3.1: Architectural Overview of the Proposed Approach for Dynamic Composition and Management of Emergency Response Processes.

Based on the given contextual information of the incident at hand, the *reasoning engine* searches the ontology and retrieves the default actions, required resources, and the response agencies responsible for performing these actions and providing the resources. The default actions are encoded in the ontology as *abstract process fragments,* which are essentially

workflows of activities that need to be performed in response to specific situations. For example, in the case of a train crash, the default actions include dispatching EMS responders to the crash site, obtaining statistical and cargo-related information (i.e., the train and wagon numbers, and the type and quantity of material in containers) from the railroad company, finding the hazard classes and risks, establishing hazard control zones and requesting transportation resources for evacuation [22]. In addition, the reasoning engine also discovers and selects the APIs of the operational systems/Web services of the relevant response agencies and resource providers based on their jurisdictions, rules, and policies.

Given the default actions (abstract process fragments) and the operational systems/service APIs of the resource provider systems identified by the reasoning engine, the *process composition modeling* component generates an integrated response process. This result process enables the IC to request resources from the selected resource provider agencies by interacting with their appropriate operational systems. Essentially, this component determines an execution order of the identified abstract process fragments and binds each activity in these fragments to appropriate operational systems and services of the response agencies. We employ a reachability analysis-based service composition approach for the generation of such an executable response process. This executable response process is then presented to the user for any customization (e.g., adding/removing some activities). The *Code generation component* of the system then generates an executable process code (e.g., in BPEL language) for deployment and instantiation on a process execution engine.

We utilize a rule-based adaptive process composition approach to handle service replacement/failure issues as well as schema changes [7]. Essentially, our process composition approach is an event-condition-action rules-based approach considering different types of events, including: 1) service failure/unavailability; 2) error messages returned by a service (e.g., due to service interface changes, input validation error); 3) change in process execution status (e.g., updated availability status of requested resources, updated threat zone); and 4) user actions (e.g., addition/removal of a response activity, making resource request to a specific response organization). Like existing works [7, 36], our approach includes

a rule base and a rule engine. The rule base includes event-condition-action rules, which define the actions for each event type. For example, failure or unavailability of service will trigger a service replacement action. Similarly, rules can be defined for compensation actions in case of service failures, for example, roll-back or cancellation of previously executed service(s) by invoking the appropriate service API. Also, there are certain types of events for which the concrete action may not be known *a priori.* Our proposed approach enables the extension and recomposition of an instantiated response process based on environmental context changes (e.g., a chemical leakage incident evolves into a fire incident) or the emergence of new requirements for such events (e.g., more fire trucks are needed). As the incident evolves, the proposed approach extends and recomposes the instantiated process in an iterative manner until the incident is resolved.

In the following subsections, we discuss each component of the proposed system in detail.

3.3.1 Emergency Management Ontology

The key component of our proposed system is the emergency management ontology, which describes the essential concepts and their relationships in the emergency management domain. The fundamental notion in the ontology is the *incident* class, which represents an emergency incident. The *incident* class has properties (i.e., address, severity, size of the affected population, potential hazards, etc.). It is associated with an *incident-type* class that describes different types of incidents (e.g., a railroad accident, a chemical plant fire, hurricane, or earthquake). Each *incident-type* is associated with: 1) *location type* (e.g., urban, rural, or industrial, etc.); 2) *default response actions*, which are represented as abstract process fragments; 3) *required resource types*; and 4) *response organizations*.

Default Actions: An abstract process fragment representing some default action is stored in the ontology as a graph of activities with partial ordering and dependence constraints. Figure 3.2 shows an example abstract process fragment of default actions for a HazMat train accident. Note that the activities in default actions are only defined at an abstract level

and lack any information about the specific operational systems for communication and coordination with the response organizations.

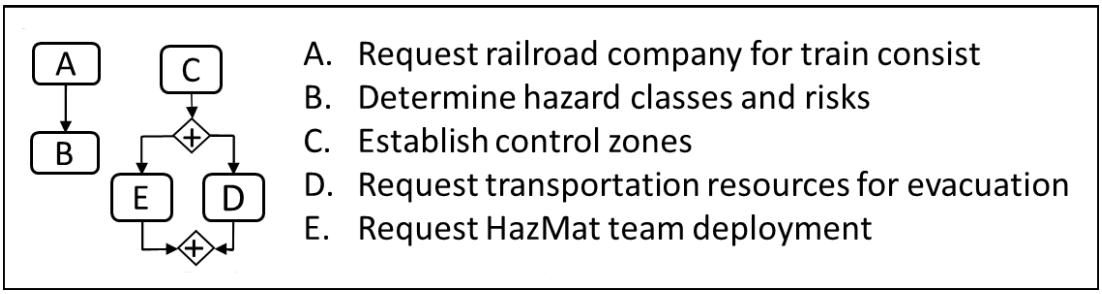

Figure 3.2: Abstract Process Fragment Showing Default Response Actions for Hazardous Materials Carrying Train Accident.

Emergency Management Resources: Emergency management resources are described in the ontology-based on National Incident Management System (NIMS) standard resource types. Each *resource* has an *owner* agency/ organization that operates in a given jurisdiction and can potentially provide the resource for assignment to an incident (e.g., a firefighting helicopter resource owned by a county fire department).

Response Organizations: *Response organization* refers to a government agency, non-governmental organization, or a private entity that is directly or indirectly involved in emergency management activities. All organizations have properties, which include its name, roles, jurisdiction (city, town, county, state, or federal), location, rules/policies, the type of resources it can provide, and the links to their operational systems/Web services APIs (WSDL files). These APIs can be used to query the response organizations' internal databases and operational systems for resource availability and commitment.

Rules: The emergency management ontology also encodes the rules of response organizations for responding to emergencies. These rules can generally be categorized into the following two types:

- *Jurisdictional Rules*: These rules specify the jurisdictions and responsibilities of the response organizations. For example, consider the following rules for determining relevant government agencies for the response process:

  - Local agencies must respond to a disaster within their jurisdiction.
  - If local agencies have exhausted their resources, then the request should be escalated to county, state, and federal-level agencies, in that order.

- *Mutual Aid Agreement Rules*: All emergencies originate at the local level; however, they can escalate, warranting the need for mutual aid from outside the affected county and contiguous counties. Mutual aid agreement rules specify the agreements between agencies for collaboration with other public and private agencies operating in the same or different jurisdiction.

  For example, the Bergen County Fire Department in New Jersey may have a mutual aid agreement rule that their department will, when requested, support firefighting operations in a neighboring county in New York State.

  There are several types of mutual aid agreements, including basic contracts between government/private organizations, local, regional, interstate, intrastate, and international agreements for assistance in the form of personnel, equipment, materials, and other associated services[1].

We encode the jurisdictional and mutual aid agreement rules of response agencies in the ontology using the Semantic Web Rule Language (SWRL). These rules are considered for identifying the response agencies, as discussed in the following section.

3.3.2 Reasoning Engine

The ontology reasoning engine is one of the core components of the system. Given the incident's contextual properties, including incident type, severity, and location, this component performs reasoning on the incident ontology to determine the default response actions, resources required, and the response organizations based on their jurisdictions and mutual aid agreements. Depending upon the incident type and characteristics, the reasoning engine is invoked to determine the default actions, resource requirements, and the potential response

[1] https://emilms.fema.gov/IS703A/RES0102130text.htm

organizations. As discussed in section 3.3.1, the default response actions are only abstract process fragments that define a set of activities that need to be performed for a particular incident type. Each activity in an abstract process fragment is realized through a composition of multiple Web services that expose the functionality of the response agencies and resource provider organizations' operational systems. The reasoning engine is responsible for the discovery and selection of relevant APIs of the response organization systems to realize activities in the selected abstract process fragments. This step involves taking into account the jurisdictional rules and mutual aid agreements between organizations.

Given the default response actions and APIs of the identified response organizations' systems, the next step is to compose a response process that enables the IC to request resources from the response agencies by interacting with their appropriate operational systems. We discuss our approach for this step in the following section in detail.

3.3.3 Process Composition Modeling

Process composition modeling involves elaborating the default actions identified for the situation at hand into a concrete response process. As discussed earlier, the default actions are only abstract workflows of response activities, which lack information about their execution order and the specific operational systems of response organizations for execution of the underlying response activities. In order to compose a concrete response process, we employ a reachability analysis-based approach that determines the execution order of the activities in the default actions and bind these activities to appropriate operational systems and service APIs.

Below we provide the important definitions followed by a detailed description of the approach.

Definition 1. *(Web service operation):* A Web service operation is defined as a 5-tuple [3],

$$s = (op\text{-}name, A_{in}, A_{out}, C_{pre}, C_{post}), \text{ where}$$

- $op\text{-}name$ corresponds to the name of the Web service operation;

- A_{in} is the set of input parameters/attributes;
- A_{out} is the set of output parameters/attributes;
- C_{pre} represents *preconditions* of s, defined with respect to the values of some attributes *before execution of* s;
- C_{post} represents *postconditions* of s, defined with respect to the values of some attributes *after execution of* s.

Our proposed approach builds on the Colored Petri net (CPN) reachability analysis-based service composition approach [29]. The idea is to explore those paths in the CPN reachability tree that satisfy a particular goal state of the system given some initial state. Such paths denote the possible execution orders of business services for response process composition. Below we first describe the CPN model and then our reachability analysis-based process composition modeling approach.

Definition 2. *(Colored Petri net)*: A Colored Petri net [29] is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$, where:

- Σ is a finite set of non-empty types, called color sets.
- P is a set of places.
- T is a set of transitions.
- A is a set of arcs, such that $P \cap T = P \cap A = T \cap A = \emptyset$. The arcs are categorized as normal arcs, read arcs, and inhibitor arcs.
- N is a node function, $N : A \rightarrow P \times T \cup T \times P$.
- C is a color function $C : P \rightarrow \Sigma$
- G is a guard function defined from T into expressions such that, $\forall t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma]$.

- E is an arc expression function defined from A into expressions such that , $\forall a \in A$: $[Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$.

- I is an initialization function defined from P into closed expressions such that, $\forall p \in P : [Type(I(p)) = C(p)_{MS}]$.

A distribution of tokens on the places is called a *marking*. An *initial marking* is determined by evaluating the initialization expressions: $\forall(p, c) : initMrkg(p, c) = (I(p))(c)$, where (p, c) denotes a token element, $p \in P$ and $c \in C(p)$.

In the response process composition context, we model the set of selected service operations as a set of transitions T in the CPN model. A transition is linked to its corresponding input and output places; each place can hold specific colored tokens. A colored token corresponds to the object class associated with the input or output parameter of a service operation. The attributes of a colored token correspond to the attributes of an object class. The preconditions of service operation are specified as guard function and the postcondition as an arc expression to the output place(s) in the CPN model. A transition can fire if its input places have the required tokens and its guard function evaluates true. The firing of a transition implies invocation of the corresponding service operation when all its input parameters are available, and its preconditions are satisfied.

Example 1 illustrates our process composition approach.

Contextual Attributes	**Initial State**
incident_type = `HazMat Train Accident'; severity =`Level 1'; location=`Urban'; city = `Saddle Brook'; state = `NJ'; latitude = '40.904009'; longitude = '74.100732'; train_id = `RAIX1102'; ...	incident_type: `not-null'; severity: `not-null'; location: `not-null'; city: `not-null'; state: `not-null'; latitude: 'not-null'; longitude: 'not-null'; train_id: `not-null'; ...

Figure 3.3: Example: Computing Initial State from Contextual Attributes

Example 1. For the train incident scenario discussed in Section 3.1, we first determine an initial state of the system by considering the contextual attributes of the incident, as shown in Figure 3.3.

To compute a goal state for process composition, we consider the system state after successful execution of activities in the retrieved default actions. For example, consider Activities A, B, and C in the default actions of Figure 3.2 for finding train consists, hazard classes & risks, and establishing control zones, respectively. Each of these activities has associated input and output attributes. For example, Activity A requires $train\_id$ as input and returns $materials$ as output; B requires $materials$ as input and returns $hazardClass$ and $risks$ as output; and C requires $materials$, $latitude$, and $longitude$ as input, and returns $safeDistances$ and $controlZones$ as output. The combined effect of the output attributes of these activities determines the goal state of the default action as shown Figure 3.4, where the output attributes $material$, $hazardClass$, $risks$, $safeDistances$, and $controlZones$ are assigned a *not-null* value in the goal state.

Default Actions	**Goal State**
A. Request railroad company for train consists B. Determine hazard classes and risks C. Establish control zones	materials: `not-null'; hazardClass: `not-null'; risks: `not-null'; safeDistances:`not-null'; controlZones: `not-null';

Figure 3.4: Example: Computing Goal State from Activities in the Default Actions.

Figure 3.5 illustrates our CPN reachability analysis-based service composition approach. Given the initial state, goal state, and the CPN models of the selected Web service operations, state-space reachability analysis is performed to determine an execution order of services that satisfies the composition goal. Figure 3.5(a) shows the CPN structures, including the places corresponding to the input/output object classes, and the transitions corresponding to the selected service operations. As shown in the figure, firing of transition t_0, which does not have any guard condition, places the tokens in the initial state in their corresponding places $Train$ and $Location$. Once the token $train\_id$ becomes available, transition t_1's guard condition is satisfied and it is fired. Thus, tokens $material$ and $qty(i.e., quantity)$

are added to the place $Train\_Consists$. The transition firing process continues until the placement of tokens (current marking) satisfies the goal state, or there are no more transitions that can be fired. Figure 3.5(b) shows the corresponding transition firing sequence for the CPN structures given in Figure 3.5(a).

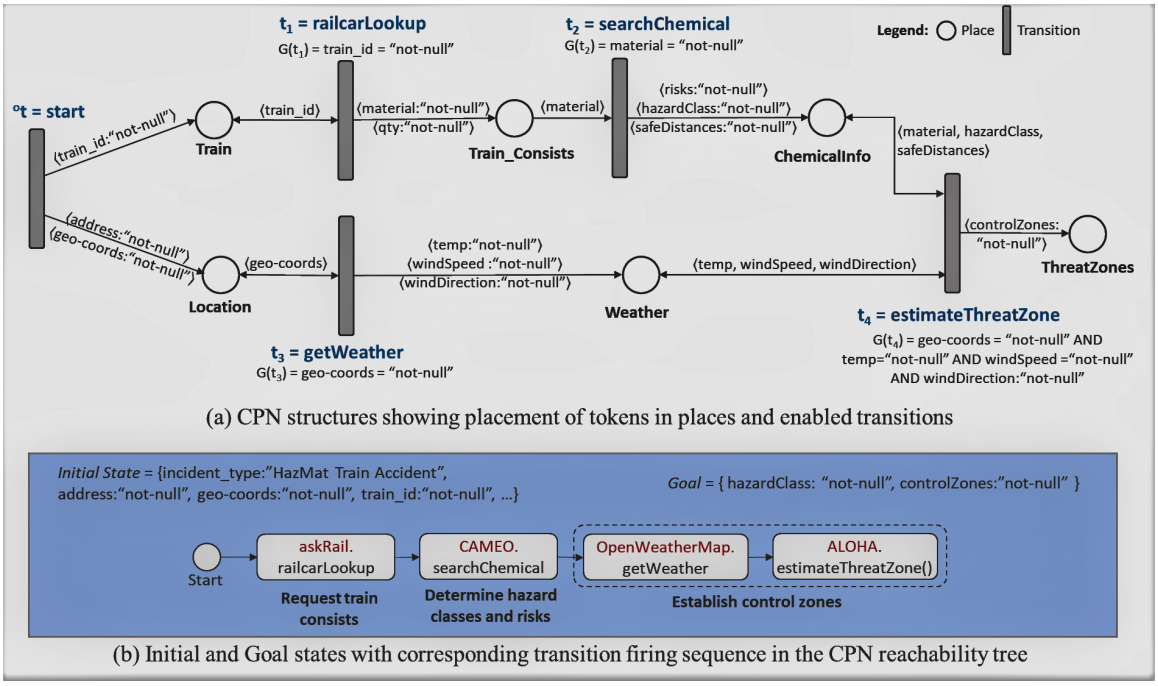

Figure 3.5: Response Process Composition

Note that we assume there is no syntactic and semantic heterogeneity between the attributes of the response activities in the default actions, and the input/output parameters of service operations of response organizations; these attributes are specified using the same terms in the default actions as well as in the appropriate Web service operations. If heterogeneity does exist between attribute names, we can employ the existing attribute-based matching approaches [3, 15] to resolve differences in attribute names before continuing with the reachability analysis.

Given the default actions determined by the ontology reasoning component, Algorithm 1 computes the workflow of the concrete response process. The algorithm takes as input, a default action $\mathcal{F}$, CPN model of the selected Web services $\in S$, and the set $C = \{\langle a_1, v_1\rangle, \ldots, \langle a_k, v_k\rangle\}$, containing the contextual attributes (a_i) with their respec-

tive values (v_i). The output is a composition of services that satisfies the goal state. The algorithm first computes the initial and goal states for required service composition in S. *computeInitialState()* function computes the state of the system from contextual attributes in C as follows:

1. All the attributes whose values are given in C are assigned abstract values, *null/not-null* or original values in case of enumerated data types.

2. All the attributes whose values are not-known are assigned *X*.

3. Relevant tokens are identified and inserted in their respective places with their values. This placement of tokens corresponds to the input marking or initial state.

ALGORITHM 1: FindServiceComposition

Input: $\mathcal{F}$: A default action

Input: CPN: Colored Petri net structures of available service operations $\in S$

Input: $C = \{\langle a_1, v_1\rangle, \ldots, \langle a_k, v_k\rangle\}$: The set of contextual attributes (a_i) with their respective values (v_i).

Output: R: Workflow of the concrete response process

1: $Init \leftarrow$ computeInitialState(C)

2: $Goal \leftarrow$ computeGoalState($\mathcal{F}$)

3: $R \leftarrow$ ReachabilityAnalysis($Init, Goal, CPN$)

4: **return** R

In the next step, the *computeGoalState()* function computes the expected state of the system after successful execution of activities in the given abstract process fragment $\mathcal{F}$ as follows:

1. Find the post-condition attributes of each abstract process fragment $\in \mathcal{F}$ by considering the expected output of component activities.

2. Assign *null*, *not-null*, or original value to the identified attributes according to the post-conditions of their respective activities.

3. Identify relevant tokens and insert in their respective places with the computed attribute values. This placement of tokens corresponds to the output marking or goal state.

Given the initial state, goal state, and the CPN model, the reachability analysis algorithm (Algorithm 2) is then invoked to compute a possible composition of selected Web services (line 3). If the reachability analysis algorithm returns a non-empty sequence of transitions, then the corresponding sequence of services is considered a valid service composition workflow R for the given default action.

Algorithm 2 provides the pseudo-code of our CPN reachability analysis algorithm, which uses the occurrence graph method proposed by Jensen et al. [29]. Given an initial state $Init$, goal state $Goal$, and the CPN model of the available service operations, Algorithm 2 performs the reachability analysis by constructing the CPN occurrence graph to find a service composition. The returned service composition is represented as a sequence of transitions fired to reach from the initial state to the goal state. If no such path exists, the algorithm returns an empty set. The algorithm maintains a queue (Q), which is a set of nodes in the occurrence graph that need to be explored for transition firing. Q is initialized as an empty set (line 1). Graph nodes are inserted in Q as new markings are generated. $createNode()$ procedure creates a new occurrence graph node from a given marking string. lines 2-6 compute the marking string from the initial state, creates the root node, and inserts in Q. The algorithm then proceeds to perform reachability analysis by extracting markings from Q as follows: An unprocessed node (m_1) is dequeued from Q, and the current marking string is extracted from m_1 (lines 7-8). Given the current marking string, $isEnabled()$ procedure checks to see which of the available transitions can be fired (lines 9-10). If a transition is enabled (all required tokens are present in input places, and the guard evaluates true), it is fired, and a new marking string m_2 is computed (lines 12-13). The procedure $isNewMarking()$ traverses the current path to see if the fired transition has updated the system state (line 14). If so, a new node is created from m_2 and inserted in Q (lines 14-16). $isEqualMarking()$ procedure compares the new marking string m_2 with

ALGORITHM 2: ReachabilityAnalysis

Input: CPN – Colored Petri net of selected service operations $\in S$

Input: $Init$ – Initial state

Input: $Goal$ – Goal state

Output: $T' = \{t_0, \ldots, t_q\}$ - Sequence of transitions fired to reach from the initial state $Init$ to the goal state $Goal$

```
1:  Q ← ∅
2:  createInitialMarking(Init)
3:  m_0 ← getCurrentMarking()
4:  createNode(m_0)
5:  parent(m_0) ← NULL
6:  enqueue(m_0, Q)
7:  while m_1 ← dequeue(Q) do
8:      setCurrentMarking(m_1)
9:      for all (transition t) do
10:         if (isEnabled(t)) then
11:             fireTransition(t)
12:             m_2 ← getCurrentMarking()
13:             setCurrentMarking(m_1)
14:             if (isNewMarking(m_2)) then
15:                 createNode(m_2)
16:                 enqueue(m_2, Q)
17:                 if (isEqualMarking(m_2, Goal)) then
18:                     return T' /* sequence of transitions fired in the current path from root to m_2 */
19: return ∅
```

$Goal$. If $isEqualMarking()$ returns true, then the graph is traversed from the root to the current node to return the sequence of transitions fired along the path, $T' = \{t_0, \ldots t_q\}$. When the goal state is reached, the algorithm returns T'. If the goal state is unreachable, the algorithm returns an empty sequence.

3.3.4 Executable Process Code Generation

Given the execution level response process fragments (service compositions) synthesized through the CPN reachability analysis-based approach, the next step is to generate an execution level workflow of the response process. Our executable process generator first identifies any control-flow dependencies between the different process fragments and then generates a workflow that includes parallel structures for all activities that can be executed concurrently. Based on the mapping information of each process activity to the appropriate operational system/service API, the code generation component then generates executable process code (e.g., in BPEL language) of the developed workflow for deployment and instantiation on a process execution engine.

Note that multiple response organizations may have independent deployments of the same system. For example, a fire department and a law enforcement agency may both use FEMA's IRIS system [23] for resource management. To reduce the state-space complexity in the CPN reachability analysis for process composition, we do not create multiple Petri net structures for same systems of different agencies. Once we identify the service composition model for a given system, we replicate that fragment in our process workflow for multiple agencies using that system.

3.3.5 Deployment, Instantiation, and Execution

In this step, the executable BPEL process generated in the previous step is first deployed on a process execution engine then instantiated and brought to execution in the runtime environment. We keep track of the process state in a collection of variables. These include: 1) user-specific variables, which correspond to user inputs and decisions taken by the user;

2) service-specific variables which includes all the elements in the service request and response messages; and 3) emergency response process-specific state information such as response activity status (e.g., assigned or completed), response provider agencies and their assignments, resource status (e.g., requested, committed, or dispatched), incident status, and events, etc. Changes in the state act as triggers for subsequent changes in the response process.

3.3.6 Dynamic Adaptation and Evolution

Depending on the execution results of the process (e.g., current resource availability status) as well as the dynamic changes in the environmental context due to incident evolution (e.g., a chemical leakage incident evolves into a fire incident) and user decisions, workflow of the instantiated response process may need to be extended to include additional activities. As the incident evolves, the proposed system extends and recomposes the instantiated response process by performing ontology-based reasoning to:

- Search for additional organizations that can provide the resources which could not be committed by organizations contacted in the previous iteration. For example, if an air ambulance is required and is not available with the local and county-level agencies, then state-level agencies may be contacted; and
- Discover additional abstract process fragments, required resources, and response agencies for any new events.

The process composition modeling and code generation components in Figure 3.1 are invoked again for executable code generation and redeployment of the extended response process. As shown in Figure 3.1, response process extension and recomposition is performed in an iterative manner. In each iteration, the current process instance is executed to completion, and based on its output and current environmental context, the process is recomposed by adding new activities and executed again. This process extension and recomposition is continued until the incident is resolved.

We utilize a rule-based adaptive process composition approach to deal with any conflicts between the already-executed part of the old process and the activities in the new composition. We assume that if there are any modifications made to the existing response process, then appropriate service APIs that allow us to perform compensatory actions (i.e., actions that would result in a partial or complete rollback and/or appropriate replacement) are available. Moreover, these compensatory actions are predefined in the rule base with respect to different events (e.g., service failure, change in the alert severity level). This is consistent with the Emergency Data Exchange Language (EDXL) messaging standards that facilitate emergency information sharing between the government entities and non-governmental organizations that provide emergency response and management services. For example, in the EDXL standard for resource management (EDXL-RM) [39], if a request for a particular resource has already been placed but as the incident evolves, we realize the requested resource is not needed anymore and a different resource is required, the earlier request can be canceled by sending a request recall message which is made available through a service API. In addition, the request for the new resource needs to be made with a service request to the appropriate agency.

3.4 Conclusion

Our integrated approach facilitates the dynamic composition of executable response processes. It develops an iterative end-to-end solution for the dynamic composition and management of an Emergency Response Management System. Such a system would provide the IC of an emergency crisis with a proposed response plan based on the disaster circumstances, location, policies, jurisdictions, and potential organizations/agencies that can perform the tasks. With changes to incident circumstances, environment, or resource availability, the proposed response plan can iteratively evolve until the disaster crisis end.

Chapter 4

Developing Personalized Patient Care Plans for COVID-19

In this chapter, we discuss how the proposed framework for dynamic composition and management of emergency response processes can be extended to develop personalized patient care plans for COVID-19 patients under home-based isolation.

4.1 Introduction

The COVID-19 pandemic is an unprecedented event in recent history that has affected the entire world. In the absence of vaccines, physical distancing, intensive contact tracing, and case isolation are considered the most effective measures to control the outbreak [34]. Healthcare facilities worldwide have been overwhelmed by the sudden influx of COVID-19 patients and are running out of both space and resources to care for existing and incoming patients. Therefore, it is crucial to keep non-critical COVID-19 patients in self-isolation at home but still provide them with proper care.

For effective management of home-based isolation, patient care should not be limited to the illness itself; it should also consider other factors related to the needs and sustenance of patients and their dependents. Some of the common factors identified for non-adherence of self-isolation include lack of financial compensation [17]; lack of support for childcare

in a single parent family [42]; comorbidities that require regular medical support outside home care [33]; and perceptions about the impact on mental health [47]. As an example, consider a patient with chronic kidney disease, requiring dialysis once a week, who is also a single mother with dependent children. Care for such a patient under self-isolation at home may require: delivery of essential medicines; food; other necessities for the patient and dependents; transportation arrangements for dialysis visits, home sample collection by trained staff for relevant laboratory tests; regular visits by a nurse; and remote monitoring of the patient condition.

A personalized patient care plan is essential to coordinate and guide the care activities for COVID-affected individuals. Traditionally, the patient care plan is developed by and implemented in collaboration with the patient, family members, caregivers, and service providers to ensure consistent and coordinated care delivery while considering the patient's condition, needs, and preferences. Ongoing care plan review is an essential requirement to support the care team to revisit the patient's health care goals and cater to the patient's evolving needs, preferences, and treatment responses to interventions [24].

A patient care plan can be considered a structured workflow of different activities requiring interaction between the various healthcare information systems, such as clinical information systems, e-pharmacy systems, laboratory testing services, and remote patient monitoring services [1]. Care plans and delivery support for patients under self-isolation at home need to be extended through integration of other service providers' systems and applications such as food delivery, medical transportation, and other essential care services. Services computing provides the critical enabling technology to facilitate coordination and information exchange among the applications and systems of caregivers and service providers for the composition and management of personalized patient care plans.

Table 4.1 shows some of the available healthcare information systems and relevant home care applications. All these systems and applications provide Web service-based interfaces to enable their integration into processes and workflows. There are also open service-based APIs for wearable devices for the use of remote patient monitoring (e.g.,

Table 4.1: Home Patient Care Systems and Services

Category	**Available Systems/Services**
Clinical information systems	NHS Care Connect Open APIs[†], Google Cloud Healthcare[†], Allscripts API for EHR integration[‡], HiPaaS[‡], Trella Health for healthcare analytics[‡]
Home health providers and Telehealth systems	Acurata Triage Scheduling Solution[‡], BetterRX and Doxy.me[‡], GetWellNetwork[‡], MatrixCare by ResMed[‡], Citus Health[‡], Home Health (infection prevention for home care and hospice)[‡], Transcend Strategy Group Solution for remote alerts and communication[‡]
Vital Signs Monitoring	Current Health's Automated Wearable Vital Signs Monitoring & Alert Escalation[‡], iHealth[‡], ManageBGL: Cloud-based diabetes management platform[‡], Fitbit activity tracker API[‡] [‡]
Emergency Transportation	Select Ambulance[‡], Ambulance Messaging - HL7 API[†], Uber Health[‡]
Pharmacy	Walgreens Pharmacy Prescription Refill[†], CVS Pharmacy API[†], H-E-B Refill Rx[‡], Truepill API[‡]
Laboratory Testing	Covid-Rapid API[†], TridentCare Portable X-Ray and Ultrasound Services[‡]
Food Delivery	Uber Eats[‡], GrubHub[‡]

[§] Open source systems, [†] Publicly available Web services, [‡] Publicly available commercial systems

heart rate, blood pressure, blood glucose levels, etc.). However, no unified system supports automatic interfacing with the operational systems of the caregivers/service providers for the development, real-time monitoring, and management of such care plans.

In this thesis, we precisely address this problem. We propose an integrated framework that enables the dynamic composition and management of personalized patient care plans. This integrated framework extends prior work on the dynamic composition of emergency response processes and applies it to the COVID-19 context [20]. Our primary objective is to demonstrate the feasibility of such an approach for creating personalized patient care plans, specifically for COVID-19 patients. The proposed framework assumes that all requisite underlying services are available through Web service-based interfaces, and that an appropriate knowledge base exists and is encoded in the form of an ontology with reasoning support.

The care plan is composed by selecting the appropriate patient care activities and integrating the systems and Web service APIs of provider organizations, as depicted in Figure 4.1. We employ ontology-based reasoning to determine the standard care activities for the given patient and identify the relevant service providers and APIs of their operational systems by considering the patient's pre-existing medical conditions and contextual knowledge (e.g., the patient's location and preferences). The discovered Web service APIs of identified service providers are then used to generate an executable care plan that is continuously monitored, updated, and recomposed based on the patient's evolving needs, preferences, and treatment responses to interventions.

4.2 Problem Statement

We address the problem of developing personalized care plans for COVID-19 patients under home isolation based on their medical conditions, needs, and requirements. The care plans are developed by integrating the Web services of the relevant caregivers and service providers into an executable workflow, which is dynamically instantiated for each patient for remote monitoring and care delivery. We state the problem as follows:

Given:

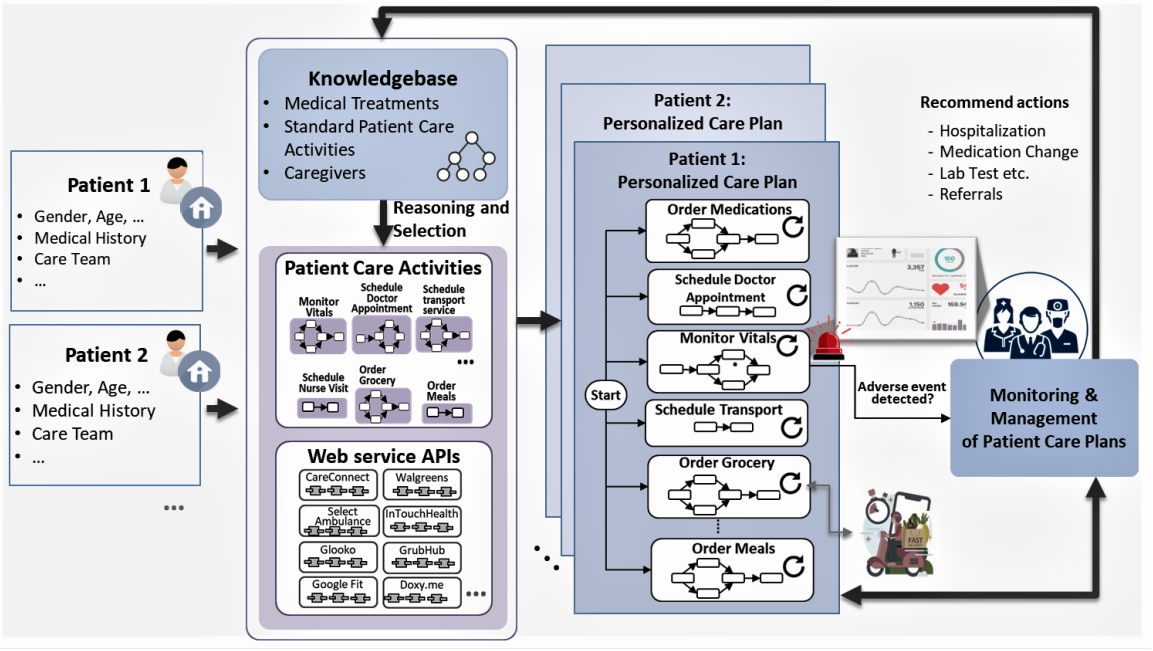

Figure 4.1: Framework for Management of Personalized Patient Care Plans

- *Individual patient information, including personal details and medical condition;*
- *Ontology, including: i) standard care activities for various medical conditions; ii) types of caregivers and service providers handling the standard care activities; and iii) resources required for the different standard care activities;*
- *Available Web services/APIs of the operational systems of caregivers and service providers.*

Develop and instantiate a personalized care plan by creating a workflow of the relevant care activities and mapping the appropriate Web services to these activities. Develop a monitoring infrastructure that can dynamically adapt and evolve the instantiated plan based on changes in the patient's medical condition and the environmental context.

4.3 Proposed Approach

The proposed framework for the dynamic composition of a patient care plan employs a multi-step approach that incrementally generates an executable care plan and enables adapt-

ability to the patient's evolving care requirements. Figure 4.1 depicts the architectural view of our proposed framework. This framework relies on an ontology for the healthcare environment and employs rule-based reasoning to dynamically develop a personalized care plan for a given patient. The ontology includes standard care activities characterized by medical conditions (both pre-existing and evolving). In addition, the ontology contains the categories of caregivers and service providers involved in at-home patient care.

Rule-based reasoning is employed to identify the required care activities based on the given patient's information (personal details and medical conditions). Appropriate APIs of the operational systems of caregivers and service providers are discovered considering the contextual knowledge (e.g., patient's location, vitals monitoring devices, insurance service, meal preferences, etc.). The standard care activities are encoded in the ontology as *abstract process fragments*, which are essentially structured workflows of activities that need to be performed for required care/service delivery. Consider the diabetic patient with chronic kidney disease who is under self-isolation at home. The standard care activities may include scheduling of weekly telehealth appointments; monitoring of patients' vitals; requesting medication refills from the pharmacy; scheduling transportation scheduling for weekly dialysis visits; and scheduling meals delivery based on patient's dietary needs. Given the care activities and the operational systems/service APIs of the service providers identified from the ontology, an abstract workflow of the care plan is generated to support care delivery. The workflow details the interaction among the operational systems of involved service providers required to provide patient care; however, the abstract workflow still needs to be converted into a concrete execution plan. Essentially, an execution order of identified care activities is determined, binding each activity to appropriate operational systems and services is performed in this step. First, we employ a reachability analysis-based service composition approach for generating an executable care plan. Then, executable process code (e.g., in BPEL language) is generated to deploy and instantiate the care plan on a process execution engine.

Patient care plans require continuous monitoring for any changes in the patient's health

conditions and evaluation of the patient's response to treatment and progress. To address this, our proposed framework includes a care plan monitoring and management component that updates the care plan whenever any new events are triggered as a result of a change in the patient's conditions or the care team's decisions.

In the following subsections, we discuss the steps of the proposed system in detail.

4.3.1 Ontology and Reasoning Support

The patient care ontology includes standard care activities for different types of medical conditions such as kidney disease, hypertension, cancer, asthma, hypertension, liver disease, and thalassemia. Each standard care activity has associated types of caregivers and service providers as well as required resources.

Standard Care Activities are high-level workflows of care activities modeled as abstract process fragments. We represent these activities using the standard BPMN. Figure 4.2 shows an example abstract process fragment of standard care activities for a self-isolated COVID-19 patient with a chronic kidney disease condition. Note that the activities are only defined at an abstract level and lack information about the specific operational systems for communication and coordination with the caregivers and service providers involved in care delivery.

Service Provider represents a caregiver organization (e.g., local health department, home health agency, non-government/private home health care organization) or any other service provider organization/individual directly or indirectly involved in patient care activities. A service provider has properties including name, service-type, and location, as well as the links to their operational systems/Web services APIs (WSDL files). These APIs can be used to query the internal databases and operational systems of the service provider for resource requests, information sharing, and service delivery.

Care Activities Selection and Service Discovery: Given the patient's profile information (including patient's personal details, medical conditions, location, vitals monitoring device information, insurance service, transportation, meal preferences, etc.) and care team's

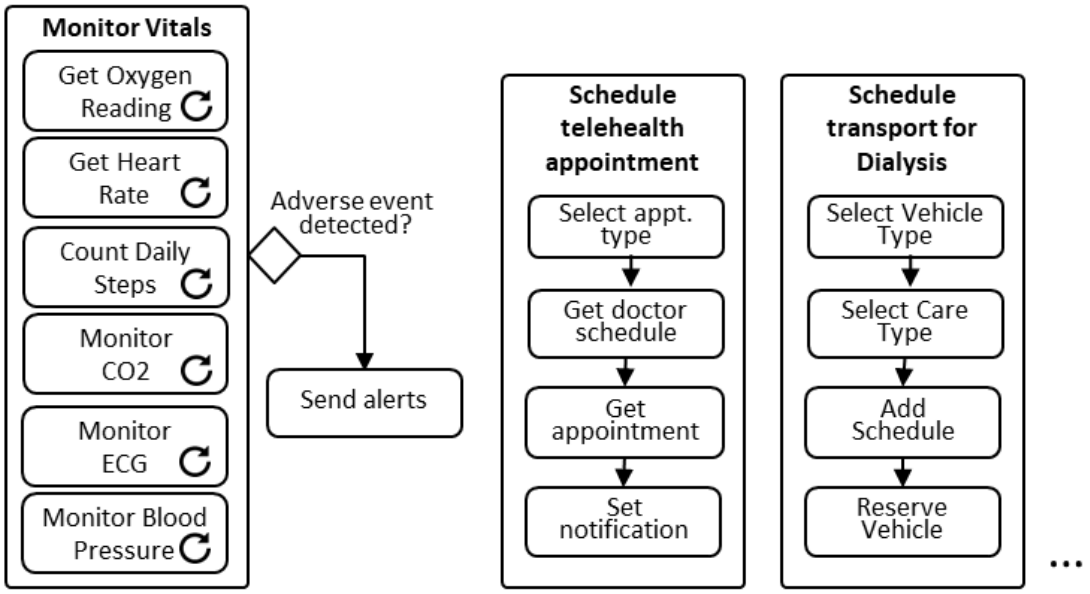

Figure 4.2: Examples of Some of the Standard Care Activities for a Self-isolated COVID-19 Patient Requiring Weekly Dialysis

```
Initial State:{
"Required Activities":
   "Remote Patient Monitoring",
   "Schedule Transport for Dialysis",
   "Schedule Telehealth Appointment",

"Gender": "Female",
"Location": "Paramus, NJ",
"Major Condition": "Kidney disease",
"Device Type": "Glucometer",…
}
```

Care Activity	Goal State Attributes
Monitor vitals:	Temperature, Pulse rate, Systolic BP, Diastolic BP, Respiration rate, Blood Sugar, …
Schedule transport for dialysis:	Repeat Frequency, Start Date, End Date, Pickup Time, Vehicle ID, Contact Details, …
Schedule telehealth appointment:	Appt Start Date, Appt End Date, Time, Appt Type, Doctor ID, Doctor Name, …

Figure 4.3: (a) Initial State Computed from Patient's Profile Attributes (b) Goal State Computed from Selected Patient Care Activities

input, the ontology reasoning identifies required care activities. In addition, ontology reasoning is employed to identify appropriate caregivers and service providers and discover their operational systems' APIs. Note that the standard care activities are only abstract process fragments. The abstract process fragment activities are realized through a composition of Web services that provide an interface to the service providers' operational systems.

Given the selected care activities and APIs of the identified service provider systems, the next step is to compose a care plan that permits the care delivery by enabling interaction and seamless integration of heterogeneous systems of the involved service providers as

discussed below.

4.3.2 Care Plan Composition

Care plan composition involves elaborating the standard care activities identified for the patient into a concrete process. As discussed earlier, the standard care activities are only abstract workflows of activities, which lack information about their execution order and the specific operational systems of service providers to execute these activities. To compose a concrete process, we employ a reachability analysis-based approach that determines the execution order of the care activities and binds these activities to appropriate operational systems and service APIs.

The general state reachability analysis problem is modeled as a 3-tuple $\langle I, A, G \rangle$ where I denotes the initial state, A denotes a set of available actions and G denotes the goal state [27]. Initial state is constituted by a set of propositions representing the initial conditions of the problem. Goal state contains a set of propositions (called problem goals). A given state consists of a set of facts. Set A consists of the actions that can be used to modify system states. Each action $a \in A$ has some preconditions denoted as prec(a) and an effect if the action is taken. The effect of an action a consists of the set of additional predicates that are evaluated to be true in the successor state (denoted by add(a)) and the set of existing predicates that are evaluated to be false and removed from the successor state (denoted by del(a)). Note that the set of predicates in add(a) and del(a) do not overlap. An action $a \in A$ is applicable to a state s if preconditions of a are satisfied by s. If an action a is applied to a state s, then the successor state s' is computed as $s' = s - \text{del}(a) \cup \text{add}(a)$. The output of the reachability analysis problem is a sequence of actions, which, if applied to the initial state I, leads to a state s such that $s \subseteq G$ [27].

In the context of care plan composition, we model the set of selected Web service operations as a set of available actions denoted by the set A. Specifically, we model each Web service operation as an available action and determine its preconditions and effects by extracting the input and output attributes from the service's description (available in

WSDL). We compute the *initial state* (I) of the composition problem from the patient's profile attributes and user inputs; the *goal state* (G) is computed from the expected outputs of selected care activities.

Example 2, detailed below, illustrates our care plan composition approach.

Example 2. Consider the example of the self-isolation patient with chronic kidney disease discussed in Section 4.1. Based on the patient's profile information and user inputs, we first compute the initial state of the process composition problem. Figure 4.3(a) shows example attributes in the patient's profile considered in the initial state. To compute a goal state for the composition problem, we consider the system state after successfully executing care activities retrieved through ontology reasoning in Figure 4.2. Each of these activities has associated input and output attributes. For example, activity *monitor vitals* requires the patient's device information as input and returns the patient's vitals, including temperature, pulse rate, respiration rate, blood sugar levels, etc. Similarly, the activity *schedule transport for dialysis* requires patient preferences as input and return scheduled vehicle information as output. The combined effect of these activities' output attributes determines the goal state of the composition problem, as shown in Figure 4.3(b).

State reachability analysis of the formulated service composition problem determines an execution order of services that satisfies the composition goal. The service execution order thus obtained for the given activities is then used to construct the workflow of the required care plan as discussed in the following subsection.

4.3.3 Code Generation, Instantiation, and Execution

In this step, any control-flow dependencies between the different care activities are identified and a workflow is generated first. Next, using the mapping information of each care activity to appropriate operational system/service API, executable process code of the developed workflow is generated (e.g., in BPEL language). This executable process is then deployed on a process execution engine and instantiated and brought to execution in a runtime environment.

4.3.4 Monitoring and Adaptation/Evolution of Care Plan

A patient care plan is a dynamic, knowledge-driven process that adapts and evolves based on changes in the patient's health conditions, circumstances, and collaborative decisions of the care team [26]. For example, a patient's medication and care needs may change based on their symptoms and medical conditions. Our proposed framework supports continuous monitoring and reviewing of the instantiated care plan by constantly evaluating patients' evolving needs, preferences, and treatment responses to interventions. Essentially, re-composition of the care plan is triggered whenever an adverse event is detected or when a member of the care team initiates a change in care plan based on some clinical decisions. Technically, this requires performing ontology-based reasoning to identify additional care activities to cater to the new requirements and invoking the process composition and code generation components for executable code generation and redeployment of the extended care plan. This care plan extension and re-composition are continued until the patient's health goals are satisfied.

4.4 Conclusion

The number of disasters that have occurred in the recent years are causing immense physical, emotional, and financial burdens. Efficient response planning can help alleviate this distress. Still, it requires communication and coordination with the diverse operational systems belonging to various collaborating government agencies, non-government organizations (NGOs), and private sector entities. Emergency response processes are not well structured, and their workflow structure and execution may evolve dynamically based on the environmental context and the type of service/activity utilized. Therefore, they cannot be statically defined for every possible situation. Instead, what is needed is adaptability to a changing situation as the incident evolves. Therefore, personalized patient care plans are necessary to ensure that each patient's needs are appropriately met. We have applied our approach as a service-oriented framework that allows dynamic composition and management

of such plans assuming the existence of an appropriate knowledge base and availability of Web-services interfaces of the underlying systems of caregivers and service providers. We developed a prototype implementation to show the feasibility of the proposed framework and discuss the challenges/issues in deploying such a system in practice.

Chapter 5

Prototype Implementation and Experimental Evaluation

This Chapter provides a brief overview of the prototype systems we have developed to illustrate the proposed framework's functionality for dynamic composition and management of emergency response processes and its application to develop personalized care plans for COVID-19 patients under home-based isolation.

5.1 Introduction

The prototype systems are built on service-oriented architecture and employ semantic-based reasoning for response process composition. We use Ontology Web Language (OWL) to create the ontologies to enable semantic-based reasoning. For the emergency response process prototype, we built on the ontology presented in Shafiq et al.'s work [45]. In the COVID-19 patients' personalized care plan prototype, we developed an initial ontology for patient care activities and related caregiver/service provider classes in OWL. Lastly, we used Apache's Jena inference subsystem for the reasoning and inference with the ontology [12]. The Web application has been developed in Java (J2EE) and deployed on an Apache Tomcat Web server. Apache ODE workflow engine has been used to deploy and execute BPEL-based code for each prototype, respectively.

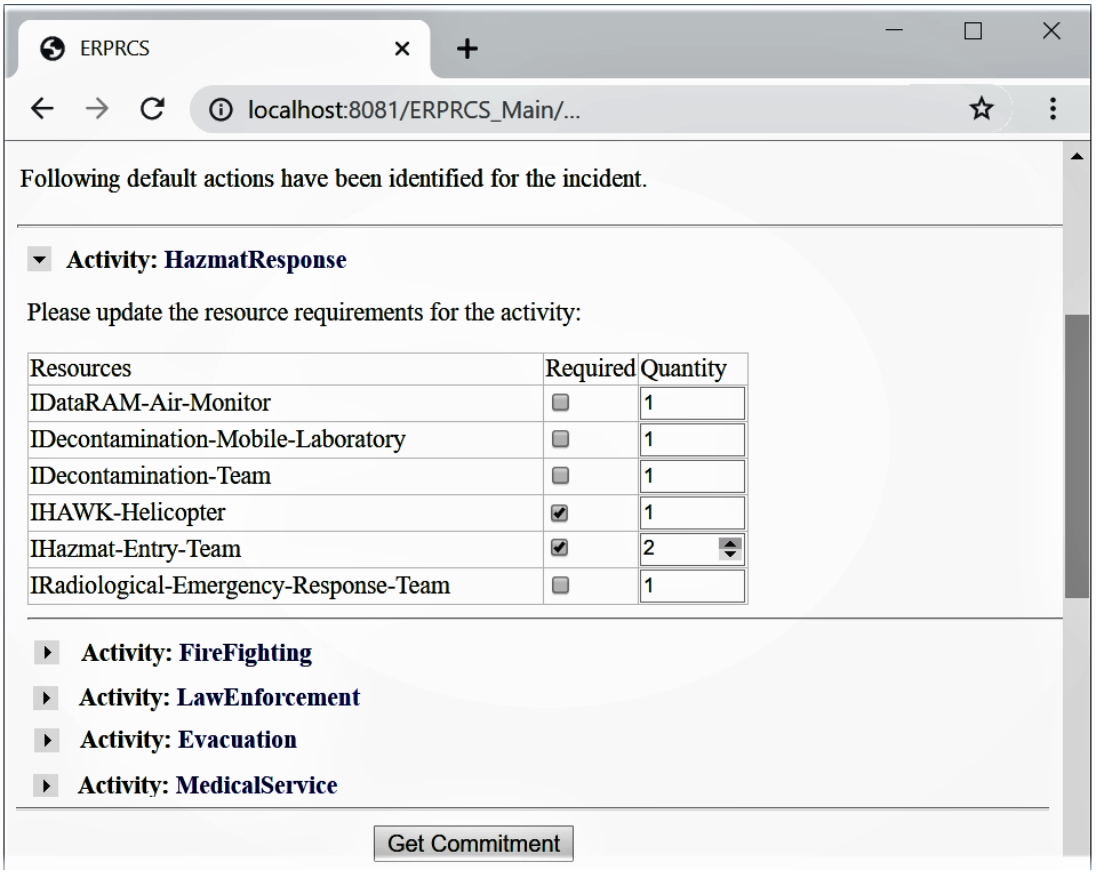

Figure 5.1: Recommended Response Actions and Resources for the Freight Train Derailment Incident

5.2 Emergency Response Processes Implementation

The user, IC, is provided an interface that facilitates the creation of a new incident in the ontology. The initial input to the system requires basic incident information as given in the incident report (i.e., incident type, date, and location, etc.). Based on this information, the system recommends default response actions along with the types of resources required, as shown in Figure 5.1. The IC can then modify the recommended actions and resources as required. Based upon the IC's decision and the information available in the ontology, the system identifies potential response organizations and the APIs of their operational systems/services for requesting the required resources. Upon the IC's confirmation, a complete response process is generated and deployed to enable interaction between the in-

cident command system and response agencies/organizations for: 1) checking availability status of needed resources; 2) making resource requests; 3) committing resources against requests; and 4) tracking the status of committed resources.

Figure 5.2 shows a partial view of the generated response process with fragments for acquiring information about the chemical hazards and assessment of any threat areas that may require rescue and evacuation operations. For example, due to a potential fire, as shown in the red block, fire engines are requested from local fire department, as shown in the blue block. Evacuation vehicles, shown in the green block, are obtained from a local private company. The right side of the figure shows the execution results of the response process fragment for the assessment of hazards and threat zones. On the left, resource commitment status is presented to the user. Thus, deploying the developed system is seamless since the proposed framework is designed to provide additional capabilities in terms of recommending the relevant response activities as well as interfacing with the relevant operational systems of the various response organizations.

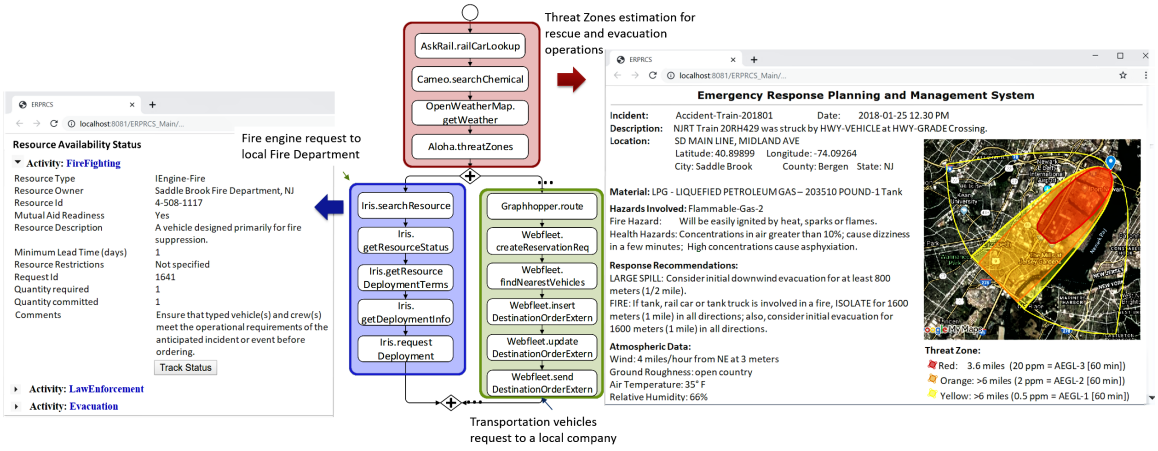

Figure 5.2: Partial View of the Generated Response Process (in the middle) and Execution Results (on left and right)

5.2.1 Experimental Evaluation

In this section, we describe the datasets used and the experiments performed to investigate the practical usability of the proposed system. We validate the effectiveness of the proposed approach using an example scenario derived from FEMA's Hazardous Materials Tabletop Exercises Manual. Specifically, we consider the scenario of hazardous chemical leakage in a train derailment incident. Response activities need to be performed based on the specifications of the scenario. Note that the abstract response activities need to be realized through concrete services provided by the operational systems of the different response organizations. For this, we need to have the specific Web service APIs that can be used to implement different tasks such as: 1) requesting resources (both material and personnel); 2) notifying commitment and deployment of resources; 3) monitoring hospital bed availability; and 4) notifying hospitals of incoming patients and injury information. However, since there are no publicly accessible APIs of operational systems used for disaster management, we simulate some of the necessary APIs and use the available systems and services related to emergency response planning.

The objective of the experimental evaluation is to show how an effective response can be composed by dynamically integrating with the systems of response organizations that may not have pre-established collaboration. To this end, we need to measure parameters such as the amount of time it takes to identify the response organizations and to integrate with their systems, as well as any limiting factors. In specific, we evaluated the following two sub-systems:

1. Ontology-based reasoning; and

2. Process composition and executable code generation.

Below we discuss the experiments and the datasets in each dimension. Note that our experimental evaluation does not consider schema changes or service failures since that is not within the scope of this work.

5.2.2 Ontology-Based Reasoning

We measure the time it takes the reasoning engine to create the new incident; identify relevant default actions, required resource types and potential response organizations; and discover the APIs of operational systems of the identified organizations.

Dataset Description: For the experimental evaluation, we created an incident ontology in Ontology Web Language (OWL) by building on the structure of the emergency management ontology presented in Shafiq et al.'s research [45]. We considered the train accident scenario presented in Section 3.1 with varying ontology domain sizes for measuring the rules inference time.

We considered the following parameters for experimental evaluation:

- **Total Number of Response Organizations**: The total number of agencies and organizations in the ontology that can potentially provide resources/services to respond to an incident on a local level, county level, state level, and federal level. We varied this number for the experiments.
- **Average Number of Resources per Organization**: The resources that individual response organizations can offer.
- **Number of Resource Types**: Total resource types encoded in the ontology.
- **Number of Identified Organizations**: Response organizations identified as potential responders through ontology rules inference.

A total of 95 resource types and five resource instances per agency were registered in the ontology.

Table 5.1 provides the dataset statistics and computation time results for the ontology rules inference experiments. The total execution time from incident creation to ontology update includes the following: 1) reading the ontology file; 2) creating a new incident

Table 5.1: Ontology Rules Inference Execution Time

Total No. of Response Organizations	20	40	80	160	320	640	1280
Avg. No. of Resources/ Organization	5	5	5	5	5	5	5
No. of Identified Organizations	9	10	33	44	44	174	464
Rules Inference Time (sec.)	6.511	8.601	8.657	8.692	15.672	36.736	157.13
Total Execution Time (sec.)	20.381	24.147	24.613	24.21	35.398	62.981	225.903

instance; 3) determining the default actions; and 4) updating the ontology and saving to the file.

The *Rules Inference Time* is the ontology rules inference time without considering file reading and saving time.

As depicted in the results in Table 5.1, the ontology rules inference time is directly correlated with the number of response organizations and resources found in the ontology. The rule inference time increases as necessary to evaluate more organizations, their jurisdictions, mutual aid agreement rules, and available resources. However, the increase is linear, and even with over 1000 organizations, requires no more than two minutes for inference and four minutes for total execution. Therefore, this is quite usable in practice.

5.2.3 Process Composition and Executable Code Generation

The objective is to measure the time it takes to generate the executable code of the response process from the abstract process fragments and response organizations' APIs identified through ontology-based reasoning. This computation includes the time to run the Petri net-based reachability analysis for service composition and the time to generate executable process code.

Dataset Description: Our experiment involved five response organizations' systems. We

Table 5.2: Available Systems and Services Related to Emergency Response Planning

Activity	**Available Systems/Services**
Resource Management	NIMS Incident Resource Inventory System (IRIS) ‡
Weather Information	Weather Web Service (Cdyne)†
Hazard Modeling	ALOHA air hazard modeling program‡
Chemicals Information	CAMEO Chemicals †
Schools Information	IES NCES Public Schools Information System †
e-Procurement	Coupa eProcurement System§, Magento§
Cargo Trains Tracking	AskRail†

§ Open source systems, † Publicly available Web services, ‡ Publicly available commercial systems

selected these organizations based on the activities in the scenario described in Section 3.1. Specifically, we have considered the following types of systems:

- **Incident Resource Inventory System (IRIS)** is provided by FEMA [23]. Various government agencies, jurisdictions, and communities use IRIS to enter inventory of resources into their databases and to share information with other agencies for incident response and mutual aid purposes.
- **Electronic Procurement System** is used for requisitioning relief goods such as medicines, water, and food supplies, from private organizations.
- **Public Schools Information System** [28] is used to query schools in affected areas to estimate the number of school children and staff for evacuation purposes.
- **Fleet Management System** is used for transportation-related resource ordering.
- **Weather and Environment Information Services** provide contextual information.
- **Cargo Train Tracking Service** is used for rail incident response organizations.

- **Hazardous Chemicals Information Services** is used to get response and and provides information about hazards, such as toxic fumes.
- **Plume Modeling System** is used to get threat zone estimates for various types of hazards.

The specific systems that we considered for performing our experiments are listed in Table 5.2.

Table 5.3: Response Process Composition Time Results

Systems	$\boldsymbol{R}_{count}$	$\boldsymbol{C}_{len}$	$\boldsymbol{T}_{comp}$	$\boldsymbol{T}_{code}$
IRIS [23]	16	5	2 sec	5 sec
Webfleet [50]	18	8	2 sec	
IES Schools [28]	20	6	22 sec	
Coupa [13]	23	10	329.33 sec	
Magento [35]	26	13	53 sec	

An important point to note here is that different government agencies and organizations which provide similar services but have different jurisdictions such as city, county, or state, often use similar operational systems (e.g., FEMA's IRIS system [23] to inventory their resources and share information with other organizations). Similarly, several private organizations often use the same e-procurement system for processing their customer orders. Even if these organizations are using different systems, there is work done on API mapping such as Afzal et al.'s work [3] that can be utilized to resolve any heterogeneity in the APIs of different organizations beforehand.

Table 5.3 provides the dataset statistics and computation time results for the process composition experiments. R_{count} denotes the total number of Web service operations modeled as CPN transitions in the given dataset. C_{len} denotes the number of service operations in a composition determined through the CPN reachability analysis algorithm (Algorithm

2) for a given default action such as the one shown in Fig. 3.5(b). T_{comp} denotes the execution time of reachability analysis-based service composition averaged over three runs. T_{code} denotes the time to generate the process workflow and executable code.

Our approach executes reachability analysis on all default actions in parallel; therefore, the overall execution time for process composition is the maximum of all the compositions. Thus, the total time for the generation of the response process is 329.33 sec + 5 sec = 334.33 sec.

Table 5.4: Coupa Requisitions API

Operation Name: $requsition\_create$
Precondition: {⟨requester≠"$null$"⟩, ⟨req_lines≠"$null$"⟩}
Postcondition: {⟨requisition_id≠"$null$"⟩}, ⟨req_status≠"$null$"⟩
Operation Name: $submit\_for\_approval$
Precondition: {⟨requisition_id≠"$null$"⟩}
Operation Name: $requisition\_show$
Precondition: {⟨requisition_id≠"$null$"⟩}
Operation Name: $requisition\_getRequester$
Precondition: {⟨requisition_id≠"$null$"⟩}

We can see that the reachability analysis time for service composition is much higher in the case of Coupa system as compared to IRIS and Webfleet. The main reason for this is that the preconditions of service operations are very well defined for IRIS and Webflee; at any given state, only a single operation can be applied. However, in the case of Coupa, the preconditions of three to four service operations are simultaneously satisfied, and because of this, the number of branches in state reachability analysis tree in Coupa are way more than in IRIS and other systems. Hence, Coupa takes more time, and an example of this

is shown in Table 5.4 [13]. We can see that upon the availability of $requisition\_id$ and $requisition\_status$ in a given state (e.g., after the execution of $requistion\_create$), the preconditions of three operations are satisfied simultaneously, resulting in the creation of four parallel branches in the reachability tree in a single state, thus incurring increased computational time.

Moreover, once the execution sequence of the Web services is determined for a default action associated with a particular organization, the determined execution sequence is added to the knowledge base for future reference to avoid the need for recomputing the reachability tree for generating response processes in future incidents.

A typical resource requisition system involves a composition of three to eight services as depicted in the case of the IRIS system. In an earlier work [3], we reviewed eCommerce and procurement business processes in several open-source ERP systems. We observed that a typical procurement order process involves three to 14 service calls mainly for resource querying, request processing, and order/dispatch confirmation.

Combining the ontology-based reasoning and reachability analysis/code generation results, we note that the generation and deployment of an executable response process may take between five to ten minutes depending on the number of response organizations involved. This time overhead for executable response process composition seems reasonable considering that the collaborating response organizations may not be known *a priori* and may not have pre-established system-level interoperability.

5.3 COVID-19 Patients' Personalized Care Plan Implementation

We demonstrate the developed prototype system's functionality using the example of a COVID-19 patient with a chronic kidney disease condition, as discussed in section 4.1.

The user (a member of the patient's care team) is provided with a Web interface that facilitates the generation and management of a personalized care plan for an individual

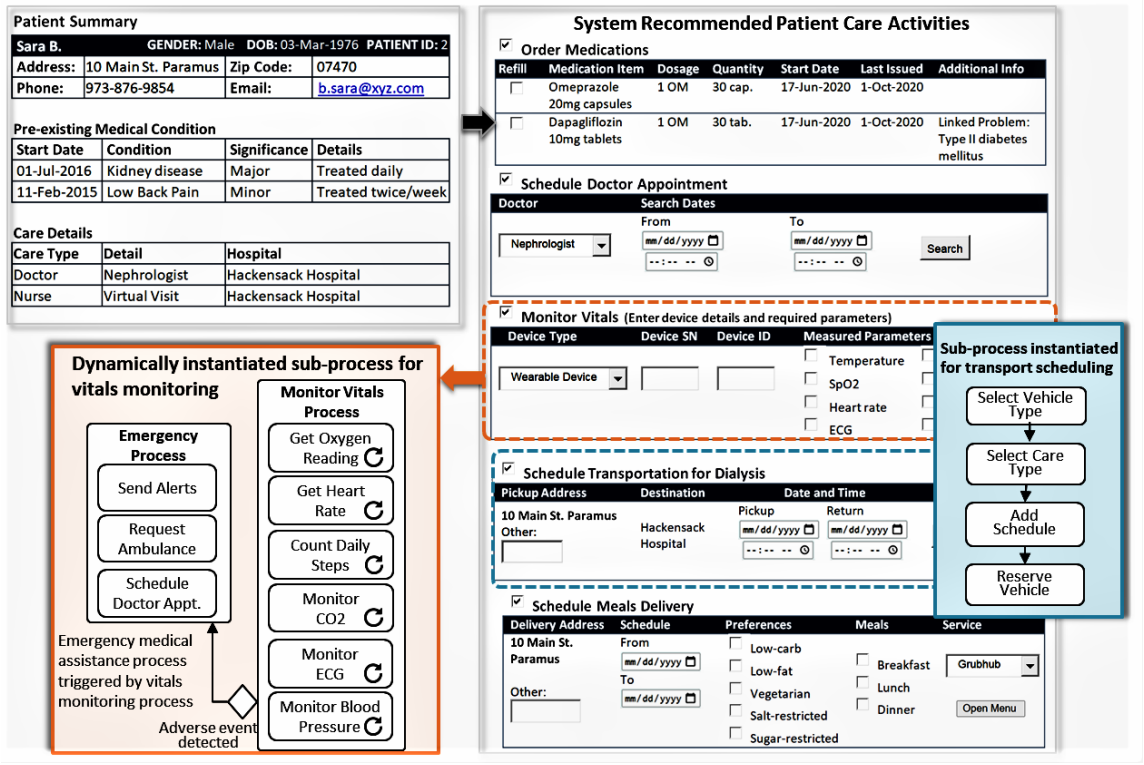

Figure 5.3: Prototype System Implementation

patient. The system's initial input requires basic information about the patient and their pre-existing medical conditions as indicated in the patient's health record. This information may include the patient's details, medical conditions, location, vitals monitoring device information, insurance service, as well as any transportation and meal preferences. This information is provided either by the patients themselves or by their caregivers or healthcare providers. For our current prototype, this information can be input through a Web interface. In general, if the system were to be deployed in a practical care setting, it should be possible to link this system to the EHR system to retrieve the requisite information automatically. Based on the given information, the system recommends standard care activities, as shown in Figure 5.3. A care team member (i.e., physician, nurse) can then review and select appropriate activities as per the patient's requirements (Figure 5.3, right panel). The team's decision and the information available in the ontology help identify relevant service providers and the APIs of their operational systems/services to execute selected activities. Upon confirmation, a complete care plan is composed and deployed, enabling interaction

between the caregivers and service providers. The formulated care plan consists of multiple sub-processes corresponding to the care plan's selected activities as depicted in Figure 5.3 for vitals monitoring, transport scheduling sub-processes, among other services.

We want to emphasize that the system supports continuous monitoring of the care plan. This monitoring allows for a reassessment of the patient's evolving needs, preferences, and treatment responses to interventions and re-composition of the care plan discussed in section 4.3.4. Members of the patient's care team can remotely monitor the patient's conditions through a Web interface and receive alerts whenever the patient outcomes deviate from the care plan's established goals.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We present an integrated approach for the on-the-fly composition of emergency response processes by enabling information sharing and interoperability among relevant response organizations' information systems. Our proposed approach does not require any pre-established collaboration among the response organizations and employs ontology-based reasoning to identify the required response activities, needed resources, and response organizations that can provide such resources. It uses the Web service APIs of these organizations' operational systems to generate an executable response process to enable interaction between the incident command system and response organizations for resource management operations. We have also experimentally validated the proposed approach's effectiveness using an example scenario derived from FEMA's Hazardous Materials Tabletop Exercises Manual. The experimental results show that the time taken to generate and deploy an executable response process is reasonable considering that the collaborating response organizations may not be known *a priori* and may not have pre-established system-level interoperability.

In the light of the current circumstances, we have studied the problem of creating personalized care plans for COVID-19 patients under home isolation. We propose a service-

oriented framework that allows for the dynamic composition and management of personalized patient care plans assuming the existence of an appropriate knowledge base and availability of Web service-based interfaces for interacting with the underlying systems. We develop a prototype to show the feasibility of the proposed framework and discuss the challenges/issues in deploying such a system in practice.

6.2 Future Work

This section presents the challenges/issues in realizing our vision in practice and then discusses the potential next steps from a technical and a policy perspective. Several technical and policy issues need to be addressed to take this system beyond a prototype level for deployment in a real-world environment. In the future, we plan to work on several underlying technical challenges, including reusability, scalability, privacy and security, interoperability, and auditability.

6.2.1 Reusability

Refinement of Default Actions/Response Plans: As presented in this thesis, our approach utilizes default action plans, incident context, and response organizations' available resources to create dynamic emergency response plans. This approach provides flexibility in generating new customized emergency response plans utilizing semantic-based reasoning and reachability analysis, as discussed earlier, which produces computation overhead. The ability to learn from emergency response plans generated for any past incidents, identification of the activities, and their associated APIs to reuse in new response plans would reduce time and effort.

As previously discussed, time is a very critical element in emergencies, and the ability to reuse these previously computed service compositions instead of regenerating the response plan afresh can significantly reduce the new response plan composition time. For

example, the response plan developed for the Saddle Brook train accident discussed in Section 3.2, involved various activities including firefighting, law enforcement, and traffic management. Each of these activities was performed by invoking the services of agencies selected based on the jurisdictional and mutual aid policies. For another incident in Saddle Brook or even at a different location, if the same activities and agencies are involved, the relevant sub-compositions of the Saddle Brook, response plan can be reused. For example, firefighting resource requests from the Bergen County Fire Department may be commonly required in the old plan as well as the plan being developed. Such reuse will save time and allow for quicker creation of emergency response plans. Our idea is to save the pre-computed response plans.

We propose to save the pre-computed response plans based on the locations and incidents to have both the ability to continue to create a response plan dynamically and to be able to develop a response plan based on past lessons learned from similar incidents. This component would provide and refine/customize the incident as per the new requirements. At completion, the IC will be provided with an interface to rate the effectiveness and accuracy of the presented and used plan. The IC, as the Subject Matter Expert (SME), would also be provided with a customized interface to refine the associated processes for the incident by adding and/or removing different actions and associated resource types, based on the location or the incident as he/she sees fit.

Response Processes Repository:

The refined learned plans created need to be saved in a well-organized repository that would allow for an intuitive, quick search for the services/processes needed for the incident at hand. The reusability of the saved, processed response processes is essential and would reduce the overhead computational time to compose an entirely new response process from scratch, which makes creating a well-indexed repository for quick and accurate search results a very critical requirement.

We propose to create a repository for existing response processes that would allow for a quick and accurate search for similar incidents within similar locations/jurisdictions in past

incidents. The repository would be indexed for retrieval of response plans, and some of the indexes might be: 1) location, which includes zip code, then city and county, where zip code would identify the area first, and the city and county would allow for better classification of jurisdictions and policies; 2) incident type and sub-type; 3) incident severity level; 4) lives, which includes whether there are there people at the incident, how many, and their age range and status; 5) health danger, which includes whether the incident provides a threat to health in the area; 6) environmental danger, which includes whether the incident provides a threat to the environment; 7) coverage/damaged area, which includes the approximate area coverage of the incident; and 8) incident accessibility, which includes whether the incident accessible by car, plane, boat, etc. This indexed repository would allow for response plans retrieval to compose new response plans based on past similar existing response plans in the system. Allowing the reusability of the categorized, processed, saved processes data would eliminate any unnecessary overhead re-computational time since agencies and APIs would rarely change.

6.2.2 Scalability

While it is easily possible for our prototype to develop personalized care plans for multiple patients, it essentially does so one at a time. However, this process becomes problematic even when only a few thousand patients are considered, so scaling it to municipalities or more significant administrative levels may cause more difficulties. One possible solution is to parallelize the underlying infrastructure and utilize appropriate load balancing solutions. Architecting such solutions is a non-trivial technical exercise. Furthermore, given the overlap in terms of patient needs, environments, and situations, it should be possible to utilize the reasoning carried out in one care plan. Essentially, it should be possible to develop a system that can take multiple care plans and appropriately split them into a set of non-overlapping components that can be individually reasoned and then reassembled to provide complete care plans for a group of patients with similar attributes and medical conditions. This is a significant technical challenge. Finally, there is a degree of human supervision

necessary in reviewing and making the final decisions on each of the personalized care plans, which can significantly increase the burden on the care team providers, especially when a multitude of plans need to be developed. It is also necessary to provide appropriate grouping/clustering solutions that can reduce the individual supervision necessary, which is another significant technical challenge.

6.2.3 Privacy and Security

Our current system assumes that there are no system-level security or privacy issues. In practice, given that all the system's underlying information is sensitive and that there are multiple participants and organizations involved in the care plan, we need to ensure that all organizational access policies and individual and legislative privacy requirements are satisfied.

There is prior research on collaborative access control [14, 30, 46] that can be applied within this context concerning organizational security. Concerning privacy, we currently assume that the system is deployed in a trusted curator setting, where the system entities looking at and operating over the data are indeed allowed to access the underlying information. If any broader insights are to be provided to policymakers, it should be possible to protect the underlying individual information through privacy models such as differential privacy [19].

6.2.4 Interoperability and Standardization

Currently, we assume no semantic heterogeneity in terms of the data/information maintained by different organizational systems or the Web service APIs. In general, this is not the case, and it is essential to resolve heterogeneity issues before the different systems can interoperate seamlessly. One such initiative was undertaken by the US Department of Homeland Security (DHS) in the form of Unified Incident Command Decision Support (UICDS) system for adopting standards-based Service-Oriented Architecture (SOA) to enable government entities, critical infrastructure owners and operators, and other private

organizations across the country to work together for an effective and coordinated response to emergencies and catastrophic situations [38, 45]. An alternative is to utilize a machine learning-based approach to resolve heterogeneity across all the different collaborating systems, as proposed by Afzal et al. [3].

6.2.5 Auditability

With a complex system consisting of multiple underlying systems, participants, and evolving environmental constraints, it is crucial to provide auditability to reduce the possibility of misuse/gamification and increase the confidence in using such a system. One possibility is to use a blockchain-based decentralized approach that essentially records each decision and the underlying justification, including the provided input and interaction between the various collaborators. This can be recorded in the form of the execution of the appropriate smart contract. Now, if a sequence of actions needs to be audited, the transaction log of each smart contract can be examined. Furthermore, using a game-theoretic approach as proposed by Akhtar et al. [4], it is possible to minimize the cost of auditing while making the overall approach incentive-compatible, thus enforcing no cheating for rational participants.

Bibliography

[1] S. S. Abidi and H. Chen. Adaptable personalized care planning via a semantic web framework. In *20th International Congress of the European Federation for Medical Informatics (MIE 2006), Maastricht*. Citeseer, 2006.

[2] N. Adam, J. Eledath, S. Mehrotra, and N. Venkatasubramanian. Social media alert and response to threats to citizens (SMART-C). In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 181–189. IEEE, 2012.

[3] A. Afzal, B. Shafiq, S. Shamail, A. Elahraf, J. Vaidya, and N. Adam. ASSEMBLE: Attribute, Structure and Semantics based Service Mapping Approach for Collaborative Business Process Development. *IEEE Transactions on Services Computing*, PP(99):1–1, 2018.

[4] A. Akhtar, B. Shafiq, J. Vaidya, A. Afzal, S. Shamail, and O. Rana. Blockchain based auditable access control for distributed business processes. In *Proceedings of the 40th IEEE International Conference on Distributed Computing Systems*, 2020.

[5] K. Amailef and J. Lu. Ontology-supported case-based reasoning approach for intelligent m-Government emergency response services. *Decision Support Systems*, 55(1):79–97, 2013.

[6] N. Assy, N. N. Chan, and W. Gaaloul. An automated approach for assisting the design of configurable process models. *IEEE Transactions on Services Computing*, 8(6):874–888, 2015.

[7] L. Baresi and S. Guinea. Self-supervising BPEL processes. *Software Engineering, IEEE Transactions on*, 37(2):247–263, 2011.

[8] A. Bucchiarone, M. De Sanctis, A. Marconi, M. Pistore, and P. Traverso. Incremental composition for adaptive by-design service based systems. In *Web Services (ICWS), 2016 IEEE International Conference on*, pages 236–243. IEEE, 2016.

[9] A. Bucchiarone, A. Marconi, M. Pistore, and H. Raik. A context-aware framework for dynamic composition of process fragments in the Internet of services. *Journal of Internet Services and Applications,* 8(1):6, 2017.

[10] A. Bucchiarone, M. Pistore, H. Raik, and R. Kazhamiakin. Adaptation of service-based business processes by context-aware replanning. In *Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on*, pages 1–8. IEEE, 2011.

[11] J. C. Buijs, B. F. van Dongen, and W. M. van der Aalst. Mining configurable process models from collections of event logs. In *Business Process Management,* pages 33– 48. Springer, 2013.

[12] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM, 2004.

[13] Coupa. Coupa Technical Documentation. `https://success.coupa.com/Integrate/Technical_Documentation/API/Resources/Transactional`, 2019 (accessed May 6, 2019).

[14] S. Das, S. Sural, J. Vaidya, and V. Atluri. Policy adaptation in attribute-based access control for inter-organizational collaboration. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC),* pages 136–145. IEEE, 2017.

[15] A. Das Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 861–874. ACM, 2008.

[16] A. De Nicola, M. Melchiori, and M. Villani. Creative design of emergency management scenarios driven by semantics: An application to smart cities. *Information Systems,* 81:21–48, 2019.

[17] B. L. Dickens, J. R. Koo, A. Wilder-Smith, and A. R. Cook. Institutional, not home-based, isolation could contain the covid-19 outbreak. *The Lancet*, 395(10236):1541– 1542, 2020.

[18] S. D¨oweling, F. Probst, T. Ziegert, and K. Manske. SoKNOS: An Interactive Visual Emergency Management Framework. In *GeoSpatial Visual Analytics,* pages 251–262. Springer, 2009.

[19] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[20] A. Elahraf, A. Afzal, A. Akhtar, B. Shafiq, J. Vaidya, S. Shamail, and N. R. Adam. *A framework for dynamic composition and management of emergency response processes. IEEE Transactions on Services Computing*, pages 1–1, 2020.

[21] R. Eshuis, R. Hull, and M. Yi. Reasoning about property preservation in adaptive case management. *ACM Transactions on Internet Technology (TOIT)*, 19(1):12, 2019.

[22] FEMA. FEMA Hazardous Materials Tabletop Exercises Manual. *FEMA Emergency Management Institute (EMI) Virtual Table Top Exercise (VTTX)*, July 2006.

[23] FEMA. Incident Resource Inventory System. `https://nimstools.preptoolkit.org/`, 2016. Accessed: 2018-04-04.

[24] C. for Disease Control and Prevention. Steps to care: Care plans. `https://www.cdc.gov/hiv/effective-interventions/treat/steps-to-care/dashboard/care-plans.html` (accessed Jan. 15, 2021), 2021.

[25] I. Georgievski and M. Aiello. Htn planning: Overview, comparison, and beyond. *Artificial Intelligence*, 222:124–156, 2015.

[26] R. E. Glasgow, A. G. Huebschmann, A. H. Krist, and F. V. Degruy. An adaptive, contextual, technology-aided support (acts) system for chronic illness self-management. *The Milbank Quarterly*, 97(3):669–691, 2019.

[27] O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas. An integrated approach to automated semantic web service composition through planning. *IEEE Transactions on Services Computing*, 5(3):319–332, 2012.

[28] IES, NCES. National Center for Education Statistics (NCES) Public Schools Directory. `https://nces.ed.gov/ccd/schoolsearch/`, 2019 (accessed May 6, 2019).

[29] K. Jensen. Coloured Petri nets: *basic concepts, analysis methods and practical use*, volume 1. Springer Science & Business Media, 2013.

[30] S. Jha, S. Sural, V. Atluri, and J. Vaidya. An administrative model for collaborative management of ABAC systems and its security analysis. In *2nd IEEE International Conference on Collaboration and Internet Computing, CIC 2016, Pittsburgh, PA, USA, November 1-3, 2016*, pages 64–73. IEEE Computer Society, 2016.

[31] M. La Rosa, M. Dumas, R. Uba, and R. Dijkman. Business process model merging: An approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 22(2):11, 2013.

[32] A. L. Lemos, F. Daniel, and B. Benatallah. Web service composition: A survey of techniques and tools. *ACM Computing Surveys (CSUR)*, 48(3):33, 2016.

[33] N. Liu, R. Huang, T. Baldacchino, A. Sud, K. Sud, M. Khadra, and J. Kim. Telehealth for noncritical patients with chronic diseases during the covid-19 pandemic. *Journal of Medical Internet Research*, 22(8):e19493, 2020.

[34] C. R. MacIntyre. Case isolation, contact tracing, and physical distancing are pillars of covid-19 pandemic control, not optional choices. *The Lancet Infectious Diseases*, 20(10):1105–1106, 2020.

[35] Magento. Introduction to the Magento 1.x REST API. `http://devdocs.magento.com/guides/m1x/api/rest/introduction.html`, 2019 (accessed May 6, 2019).

[36] A. Marrella, M. Mecella, and S. Sardina. Intelligent process adaptation in the SmartPM system. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):25, 2017.

[37] J. Moreira, L. Ferreira Pires, M. van Sinderen, and P. Costa. Towards ontology-driven situation-aware disaster management. *Applied ontology*, 10(3-4):339–353, 2015.

[38] J. W. Morentz. Unified incident command and decision support (UICDS): a Department of Homeland Security initiative in information sharing. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 321–326. IEEE, 2008.

[39] OASIS. Emergency Data Exchange Language Resource Messaging (EDXL-RM) 1.0. OASIS Standard. *OASIS Emergency Management Technical Committee*, December 2009.

[40] G. Redding, M. Dumas, A. H. Ter Hofstede, and A. Iordachescu. Modelling flexible processes with business objects. In *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 41–48. IEEE, 2009.

[41] M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive process management with ADEPT2. In *21st International Conference on Data Engineering (ICDE'05)*, pages 1113–1114. IEEE, 2005.

[42] S. Rizvi Jafree, S. A. Naqi, et al. Significant other family members and their experiences of covid-19 in Pakistan: A qualitative study with implications for social policy. *Stigma and Health*, 5(4):380, 2020.

[43] L. Sabatucci and M. Cossentino. Supporting dynamic workflows with automatic extraction of goals from BPMN. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 14(2):1–38, 2019.

[44] C. Sell and I. Braun. Using a workflow management system to manage emergency plans. In *In Proceedings of the 6th International ISCRAM Conference*, volume 41, page 43. ISCRAM, 2009.

[45] B. Shafiq, S. Ae Chun, V. Atluri, J. Vaidya, and G. Nabi. Resource sharing using UICDS framework for incident management. *Transforming Government: People, Process and Policy*, 6(1):41–61, 2012.

[46] B. Shafiq, J. B. Joshi, E. Bertino, and A. Ghafoor. Secure interoperation in a multidomain environment employing rbac policies. *IEEE transactions on knowledge and data engineering*, 17(11):1557–1577, 2005.

[47] L. E. Smith, R. Amlot, H. Lambert, I. Oliver, C. Robin, L. Yardley, and G. J. Rubin. Factors associated with adherence to self-isolation and lockdown measures in the UK: a cross-sectional survey. *Public Health*, 187:41–52, 2020.

[48] W. Song and H.-A. Jacobsen. Static and dynamic process change. *IEEE Transactions on Services Computing*, 11(1):215–231, 2016.

[49] P. Tang and G. Shen. Decision-making model to generate novel emergency response plans for improving coordination during large-scale emergencies. *Knowledge-Based Systems*, 90:111–128, 2015.

[50] T. Telematics. WEBFLEET.connect 1.43.0 Reference. `https://telematics.tomtom.com/en_gb/webfleet/partners/integration/developer-resources/`, 2019 (accessed May 6, 2019).

[51] W. M. Van Der Aalst. Configurable services in the cloud: supporting variability while enabling cross-organizational process mining. In *On the Move to Meaningful Internet Systems: OTM 2010*, pages 8–25. Springer, 2010.

[52] H. Wang and Q. Zeng. Modeling and analysis for workflow constrained by resources and nondetermined time: An approach based on Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 38(4):802–817, 2008.

[53] J. Wang. Emergency healthcare workflow modeling and timeliness analysis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(6):1323–1331, 2012.

[54] J. Wang, W. Tepfenhart, and D. Rosca. Emergency response workflow resource requirements modeling and analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(3):270–283, 2009.

[55] J. Yu, Q. Z. Sheng, J. K. Swee, J. Han, C. Liu, and T. H. Noor. Model-driven development of adaptive web service processes with aspects and rules. *Journal of Computer and System Sciences*, 81(3):533–552, 2015.