The Report Committee for Melissa Mei Yun Lee
Certifies that this is the approved version of the following Report:

Modular Supervisory Controller for Complex Systems

APPROVED BY
SUPERVISING COMMITTEE:

Eric van Oort, Co-Supervisor

Mitch Pryor, Co-Supervisor

Modular Supervisory Controller for Complex Systems

by

Melissa Mei Yun Lee

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2018

Acknowledgements

I would like to thank my advisors, Dr. Eric van Oort and Dr. Mitch Pryor, mentors, family, and friends for their unwavering guidance and support.

Abstract

Modular Supervisory Controller for Complex Systems

Melissa Mei Yun Lee, M.S.E.

The University of Texas at Austin, 2018

Supervisors: Eric van Oort, Mitch Pryor

Automation for the oil and gas industry is driven by the need to improve efficiency, productivity, consistency, and personnel safety, while reducing cost. Fully automated systems alleviate the physical toll on human operators and allow them to focus on monitoring unsafe well events and machinery maintenance. Complex systems like drilling rigs and snubbing units require supervisory controllers that can safely coordinate equipment and processes, overcome interoperability challenges and allow for functional scalability without sacrificing safety, security, and consistency of operations. The primary objective of this report is to explore the feasibility of developing a modular supervisory controller architecture which addresses these concerns by modifying and extending existing architectures. Such modifications include the use of non-homogeneous models in sub-system modules, including discrete event models for control and physics-based models for collision avoidance, addition of a system compilation module (Meta Module) to identify simple design errors, and implementation of an algorithm for synthesis of modules and filters to replace missing sub-systems. This report discusses the implementation results of the modular supervisory control architecture (modMFSM) on a

simplified two-machine drilling system for assessment of design practices. Simulations for three test cases were executed to assess the ability of the controller to correctly perform error-free operations, detect and react to possible collisions, and adapt to missing equipment. The report then discusses the possibilities of extending the modMFSM architecture to control large complex systems such as drilling rigs, using snubbing operations as an example.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Drilling Automation Background

Automation for industrial applications, such as in the manufacturing and the oil and gas industries, is driven by the need to improve efficiency, productivity, consistency, and personnel safety, while reducing cost (Kurz, 2013). In the oil and gas industry, there is ongoing effort in developing mechanized and semi-automatic rig equipment, and control systems, supported by high quality instrumentation (Macpherson et al., 2013). This technology not only allows many routine drilling operations to be remotely operated from the driller's cabin rather than manually on the rig floor, thereby improving personnel safety, but also provides the driller with optimal drilling parameters suggestions, dysfunction monitoring and alarms, and regulation of drilling parameters within a safety envelope, increasing consistency of performance and drilling dysfunction prevention (Macpherson et al., 2013). The advent of commercially available mechanized and semi-automatic drilling rig systems has initiated a surge of effort to obtain fully automated rigs, which require supervisory controllers that act in the capacity of the driller, coordinating rig equipment for execution of drilling activities. Fully automated rigs alleviate the physical toll on the rig crew and allow them to focus on monitoring unsafe well events and machinery maintenance, while being taken out of harm's way.

Due to financial obstacles including hardware acquisition, retraining, and the extended life-cycle for rigs and rig equipment, automation efforts have been limited. Commercially available supervisory controllers such as the NOV Operating System (NOVOS) that allow for fully automated performance of repetitive drilling activities, such as tripping, drilling, and pipe connections are currently in development (National Oilwell Varco, 2018b). However, these controllers are limited by their lack in modularity

and ability to accommodate for varying combinations of equipment from multiple sources. This challenge is a major obstacle for development and implementation of fully automated drilling rigs (Macpherson et al., 2013).

1.2 Supervisory Control Motivation

The motivation for this research was the need to develop a supervisory controller architecture for snubbing operations. Snubbing operations are a subset of drilling, completions, and production operations, and are performed when it is necessary to run pipe or other tubulars into or out of a pressurized well for activities such as tripping, underbalanced drilling, milling, fishing, and well control operations (Grace, 2003). The main benefit of snubbing is the ability to work in a high pressure well without pumping high weight fluids or mud into the wellbore, reducing the risk of damaging the formation (Grace, 2003). Snubbing has three main modes of operation: pipe-light, pipe-heavy, and balance point (Prebeau-Menezes, 2013). Pipe-light operations are needed when the force of the pressurized well exceeds the weight of the drill string, which includes the drill pipes, drill collars, various tools, and the drill bit. In pipe-light situations, if downward force is not constantly applied to the pipe, forces from the well could expel the drill string from the well (Prebeau-Menezes, 2013). Pipe-heavy operations are performed when the weight of the drill string exceeds the force of the pressurized well. Balance point operations are performed to safely transition the well between pipe-light and pipe-heavy operations when the well force approximately equals the weight of the drill string (Prebeau-Menezes, 2013).

Snubbing units can either work as a standalone unit, or as a rig-assist unit, which is an add-on to a drilling rig. Equipment on standalone and rig-assist units are similar (Prebeau-Menezes, 2013). Appendix A – Snubbing Unit Equipment describes general

equipment found on snubbing units. In a traditional snubbing unit, the snubbing crew works in a work basket located directly above the wellbore. Working in the work basket while the well is under high pressure is extremely dangerous, especially during pipe-light operations, during which an expelled pipe could critically injure the crew in the work basket (Prebeau-Menezes, 2013).

Superior Energy Services, Inc. has developed mechanized snubbing rigs which allow for remote-control of snubbing equipment and catwalk from an adjacent rig cabin, removing crew from the rig floor for routine operations (Superior Energy, 2016). Appendix D – Process Narrative Flowcharts for Snubbing presents a detailed process narrative of a Superior mechanized snubbing rig. This state-of-the-art technology has made significant improvements to snubbing operation safety, and consistency. However, many of the repetitive operations such as snubbing in/out tubulars, stripping in/out tubulars, and pipe handling during pipe-light and pipe-heavy operations can be fully automated by introducing a supervisory controller to send commands and set points to the mechanized equipment. Using a supervisory controller to implement repetitive operations not only further improves performance consistency and safety but also frees the snubbing crew to monitor performance and watch for dysfunctions.

Snubbing operations are closely related to drilling operations. The smaller operations scope of snubbing and the existence of commercially available, mechanized snubbing units make snubbing operations a perfect candidate as a platform for developing and testing a modular supervisory controller for use in the oil and gas industry.

1.3 Problem Statement & Research Objective

In order to automate systems like drilling rigs and snubbing units, there is a need for a supervisory controller that enables easy integration and safe coordination of

different automation products. There are a variety of approaches to supervisory control, reviewed in Section 2.2 Supervisory Control Techniques Overview, but for most individual components such as rig equipment, the well-understood Finite State Machine (FSM) is sufficient; however, this approach is untenable when considering the coordination of all equipment. As will be shown below, other supervisory controllers found in the literature have challenges that need to be addressed prior to application to drilling automation in the oil and gas domain. Thus, this report will evaluate the feasibility for use of a modified version of the modular finite state machine architecture (modMFSM) with modules developed for reconfiguration and environment interaction abilities through implementation on a simplified conveyor belt drilling system. This report will then discuss the extension of the modMFSM architecture to drilling automation with the context of application to snubbing operations. This approach holds promise to address modularity and scaling issues, as well as allow use of modules encompassing various model structures, and enable automatic generation of modules and filters for human intervention when missing equipment.

The hypothesis of this research is stated below:

It is feasible to develop a modular, supervisory controller that allows for modularity, functional scalability, safe interaction with the environment, and controller reconfiguration for complex systems with complicated operations, well-defined tasks, and interdependent equipment.

As shown in the literature review, there is a need for development of a modular supervisory controller for complex systems, such as drilling rigs. Controllers for these systems must allow for functional scalability without sacrificing safety, security, and consistency of operations. As a proof of concept for testing of the modular supervisory

controller, a study of implementation on a simplified conveyor belt drilling system is performed. In summary, the objectives of the project were to:

1. Develop a supervisory controller architecture, modMFSM, that addresses the critical concerns described above.
2. Implement the developed solution of a simplified, but representative system.
3. Demonstrate the controller's ability to detect potential collisions, and accommodate equipment replacement.
4. Present extension of the controller architecture for implementation for snubbing operations.

Chapter 2: Literature Review

This section first provides a review of the state-of-the-art supervisory control as used in the oil and gas industry, after which an overview of supervisory control techniques is presented.

2.1 Supervisory Control in the Oil and Gas Industry

Implementation of supervisory control in the upstream oil and gas industry began with control and monitoring of production activities in oil fields (De, Silin, & Patzek, 2000; Dunham, 1987; Niven, 1971; Wilson, 1971). These supervisory systems were strictly serial and had slow response times, since "events in the oilfield [do not] occur at such a rate that requires immediate computer response" (Wilson, 1971). Since the implementation of these supervisory systems, the role of a real-time control system user in the oil and gas industry has been "moving away from [manual control in which one] … actively monitors the state of a system, identifies when there is the need for control input, and takes the necessary action to ensure process parameters remain where they are expected to be" toward a role in which one's "main function is to monitor the automation and to be ready and able to intervene – to re-take manual control – should problems arise with the automated systems" (McLeod, 2015).

In the late 1980s, Supervisory Control and Data Acquisition (SCADA) Systems were developed for monitoring and control of distributed processes where a master terminal unit (MTU) controls remote terminal units (RTU) that interact directly with the lower level process controllers (Gaushell & Darlington, 1987; Ito, 1997). The challenges of SCADA systems are that these systems are best suited for environments with "stable ambient conditions," maintaining reliable communication and calibration is often difficult, and there is sensitivity to power fluctuations (Ito, 1997). In addition to SCADA

systems, development of mechanized rig equipment for teleoperation began to allow remote control of certain drilling operations away from the rig floor; this mechanization effort has led to significant progress in development of tele-operated and semi-automatic operations (Brugman, 1987; Cao & Moralez, 2016; US5988299 A, 1999; C. W. Lee & Won, 2013; Loeyning, 2017; Schlumberger Limited, 2018; Ugasciny, Chang, & Hampson, 2016).

Maturation of increasingly sophisticated mechanized and semi-automatic rig systems spurred interest in development of fully automated drilling rigs. Some of the major cutting-edge efforts in the industry are detailed in (Ayling, Jenner, & Neffgen, 2003; Calderoni & Cercato, 2015; Huisman, n.d.; Jacobs, 2015; MacGregor, n.d.; Nabors, 2018; Nabors Industries Ltd., 2018; Ornas, 2010; West Group, 2015). The surge of effort in drilling automation has also led to the development of supervisory control systems for automatic execution of drilling sub-operations through employment of automata-like network of states shown in (National Oilwell Varco, 2018b), and Petri nets (Prati, Farines, & de Queiroz, 2015). Saadallah developed a DES model of a drilling control system for coordination of drilling equipment and activities as a Petri net (Nejm Saadallah, 2013). Simulations were done to show the feasibility of this control model to be implemented for either semi-automatic or fully automatic control of drilling activities. NOVOS, a commercially available drilling system with controllers for automatic execution of drilling activities is in the process of being expanded to allow for fully automatic coordination and execution of drilling activities (National Oilwell Varco, 2018a).

In order to fully automate large systems in complex environments such as those found in the oil and gas industry, development of supervisory controllers which can safely coordinate various equipment and processes are critical. Supervisory control

efforts in the oil and gas industry have significantly progressed from automated monitoring and control of relatively stable processes in oil fields to development of automated drilling processes, which require significantly quicker responses in more unpredictable environments. Despite the advancements of supervisory control development for drilling in the oil and gas industry, many issues remain to be tackled, including those of modularity, and functional scalability, which motivate the need to explore the development of a modular supervisory controller architecture for drilling operations and other systems of similar size and complexity.

2.2 Supervisory Control Techniques Overview

Supervisory control is used in applications for which "control and coordination" are required for "orderly flow of events" (Ramadge & Wonham, 1989). Supervisory controllers should satisfy safety requirements, enable task planning, allow for functional scalability, and be verifiable (N. Saadallah, Meling, & Daireaux, 2011). Discrete event systems (DES) have discrete state spaces that adapt to "abrupt occurrence, at possibly unknown irregular intervals, of physical events" (Ramadge & Wonham, 1989). DES can be modeled as untimed, timed, and hybrid systems. Untimed system models are purely event-driven and do not rely on time information. Timed system models are ones where timing information is crucial. Hybrid systems "combine time-driven with event-driven dynamics" (Cassandras & Lafortune, 2011). Recent research on hybrid systems have applied them for real-time manufacturing operation as in (Saez, Maturana, Barton, & Tilbury, 2018) and analyzed approaches for inclusion of fault detection and isolation as in (Khorasgani & Biswas, 2018). The routine operations of the systems considered in this report are assumed to be event driven with negligible dependence on timing information. Therefore, only DES control techniques will be explored. Although many different types

of DES models exist, the more common DES models explored in this literature review are finite state machines (FSMs), Petri nets, and Markov decision processes.

2.2.1 Finite State Machines

A common approach to modeling DES is through the use of finite state machines (FSMs). Deterministic FSMs are represented by the six-tuple $G = (X, E, f, \Gamma, \mathrm{x}_0, X_m)$, where

X = finite set of states

E = finite set of events

f = transition function that maps starting states and events to ending states

Γ = active event function to define the events that allow transitions

x_0 = initial state

X_m = finite set of marked states

There are two basic variants of FSMs: Moore machines which produce outputs as a result of the current state, and Mealy machines which produce outputs as a result of the transition between states (Cassandras & Lafortune, 2011). In order to capture instances when transition allowance depends on external variables such as time, guards—generally in the form of if-statements—are added to the transition definitions. Guarded FSMs are called extended finite state machines (Fowze & Yavuz, 2016). In traditional automata theory, supervisory controllers control FSM systems by disabling undesired transitions (Cassandras & Lafortune, 2011)]. In order to ensure desired system behavior, the widely used software tools, such as SMV and SPIN, among others, can be used for model checking of FSMs.

FSMs generally display good performance for small and medium scale systems. FSMs "can immediately access the control pattern for each controllable transition based on the current encoded state," and can perform reachability searches, assuring prompt controller response, which is crucial for systems with strict real-time requirements (Zhu & Brooks, 2009). However, this quick response has a high computational cost dependent on the size of the state space. There is potential to mitigate this cost through separation into sub-systems that limit the search to the state space of the relevant sub-system.

2.2.2 Petri Nets

Using a Petri net is another common approach to modeling DES, and was the chosen approach in Saadallah's drilling control system model (Nejm Saadallah, 2013). Petri nets have also been used for hybrid systems, and modular supervisory control (Basile, Chiacchio, & Coppola, 2012; J. S. Lee, Zhou, & Hsu, 2007; Lennartson, Bengtsson, Wigström, & Riazi, 2016; Nishi, Watanabe, & Sakai, 2018; Wu, Zhou, & Chu, 2008). A Petri net is represented by the tuple $S = (P, T, I, O, U)$, where

P = finite set of places, which can hold tokens for state-specific requirements

T = finite set of transitions

I = finite set of arcs from places to transitions

O = finite set of arcs from transitions to places

U = integer vector representing the current marking

The state of the Petri net is shown by its marking, which is a "vector expressing the number of tokens in each place" (Zhu & Brooks, 2009). Conditions on transitions in Petri nets can be modeled with predicate / transition Petri nets for meeting conditions

before transitioning (Genrich, 1986). SMV and SPIN cannot be used for model checking Petri nets directly. Methods exist for translation of Petri nets to language used in the mainstream formal model checking software like SMV; however, use of finite state machines would avoid potential errors that could be lost in translation (Szpyrka, Biernacka, & Biernacki, 2014).

Compared to FSMs, Petri nets have greater modeling power, allow for efficient real-time control and analysis due to mathematical computation of properties which employ linear matrix algebra, and allow for automatic handling of concurrent events (Zhu & Brooks, 2009). "An automaton can always be represented as a Petri net, [but] not all Petri nets can be represented as finite-state automata; [therefore], Petri nets can represent a larger class of languages" (Cassandras & Lafortune, 2011). Petri nets also provide more compact state spaces than FSM, making them better suited to model systems with repeated structure (Zhu & Brooks, 2009). Additionally, Petri nets are capable of modeling concurrent models in a simpler format than FSMs; however, more problems are decidable for FSMs than for Petri nets (Cassandras & Lafortune, 2011). The main trade-off between FSMs and Petri nets is model richness versus decidability.

2.2.3 Markov Models

Markov models are stochastic models that capture environments with probabilistic transitions. In order to be modeled by Markov models, the environments must be proven to obey the Markov property, which states that the value of the next state depends solely on the current state and not the path history that led to the current state.

Although Markov models are useful for capturing uncertainty, they are data intensive and require knowledge of transition probabilities, rewards for each state, action,

and observation. They also require “every attribute value combination to be enumerated,” which can lead to large state spaces for small problems (Cassandra, 1998).

2.2.4 Selection of DES Models

FSMs, Petri nets, and Markov models all have their advantages and disadvantages. The choice of which approach to use for control of a DES is specific to the application. Markov models depend heavily on knowledge of the environment that may not be available for all systems, which would then require learning techniques. Therefore, Markov models were not used for modeling the type of systems explored in this paper. In choosing between FSMs and Petri nets, the main point of consideration is in model richness versus decidability. Since Petri nets can be derived from FSMs should FSMs prove to be unable to provide the level of model richness necessary for controllers of the complex systems explored in this paper, FSMs were used in this controller architecture to allow for better decidability.

2.2.5 Modular Approaches Using Finite State Machines

One of the main disadvantages of modeling systems as FSMs is the tendency for the state space to grow rapidly for complex systems, which lead to exponential number of states and processes. However, this state space explosion can be mitigated through modular controller synthesis by “decomposing the [the system] into simpler components…[which allows] greater structure and flexibility to be incorporated into the controller” (Ramadge & Wonham, 1989; Thistle, 1996).

Endsley proposed a particularly promising modular supervisory control architecture, which, devised for complex manufacturing systems, decomposes the systems into sub-systems housed in structures called modules which interact with each

other to form a modular supervisory controller (Endsley, 2004). Since input / output finite automata called trigger / response finite state machines (TR FSMs) are used to model the subsystems inside the modules, this architecture is called Modular Finite State Machines (MFSMs). Unlike traditional finite state machine supervisory controllers which block illegal actions of the plant, the MFSM supervisory controller "forces events to occur" (E. E. Almeida, Luntz, & Tilbury, 2007). Defined in this section are the major components of the MFSM architecture.

TR FSMs used in MFSM theory are defined by $\mathrm{FSM} = (\mathrm{X_{FSM}}, \mathrm{T_{FSM}}, \mathrm{R_{FSM}}, \tau_{\mathrm{FSM}}, \mathrm{x_{0_{FSM}}}, \mathrm{x_{dp_{FSM}}}, \mathrm{X_{M_{FSM}}})$, where

$\mathrm{X_{FSM}}$ = finite set of states in the FSM

$\mathrm{T_{FSM}}$ = finite set of triggers recognized by the FSM

$\mathrm{R_{FSM}}$ = finite set of responses produced by the FSM

τ_{FSM} = finite set of transitions between states in the FSM

$\mathrm{x_{0_{FSM}}}$ = initial state

$\mathrm{x_{dp_{FSM}}}$ = dump state (state without outgoing transitions)

$\mathrm{X_{M_{FSM}}}$ = set of marked states (accepting states)

Dump states are generally not shown in the TR FSM diagram unless there exists a transition to that state.

Unlike in traditional automata theory, the supervisory controller in MFSM theory is designed to only include desired state transitions rather than disabling actions of the DES. Therefore, transitions of TR FSMs are defined by $\tau = (\mathrm{t_\tau}, \mathrm{x_{s_\tau}}, \mathrm{x_{d_\tau}}, \mathrm{r_\tau})$, where

$\mathrm{t_\tau}$ = trigger for transition

x_{s_τ} = source state of the transition

x_{d_τ} = destination state of the transition

$r_\tau = [r_{\tau,1}, \dots, r_{\tau,m_\tau}]$ = set of responses as a result of the transition

As previously mentioned, TR FSMs in MFSM theory are housed in modules. The modules communicate with each other through ports on the modules called sockets, which are defined by, $S_{M_i} = (T_{M_i}, R_{M_i})$, where

T_{M_i} = set of triggers that travel in through socket S_{M_i}

R_{M_i} = set of responses that travel out through socket S_{M_i}

Therefore, modules can be defined by the TR FSM they house and the sockets through which they communicate with other modules. A module is defined by $M = (FSM, S_M)$, where

$FSM = (X_{FSM}, T_{FSM}, R_{FSM}, \tau_{FSM}, x_{0_{FSM}}, x_{dp_{FSM}}, X_{M_{FSM}})$

$S_M = \{S_{M_1}, \dots, S_{M_n}\}$ for n number of sockets

Filters act as the interface between two modules, defining the interaction between them. Therefore, filters house classical FSMs with no responses. Filters are especially important in the execution of modular verification because modules that satisfy the specifications of their filters also satisfy the specifications on their interaction with other modules. This type of verification for modular systems is similar to contract-based design, which is commonly used for development of component-based software design and multi-component cyber-physical systems (Nuzzo, Finn, Iannopollo, & Sangiovanni-

Vincentelli, 2014; Söderberg & Johansson, 2013). Filters are defined by $F = (FSM, S_{F_a}, S_{F_b})$, where

$FSM = (X_{FSM}, T_{FSM}, R_{FSM}, \tau_{FSM}, x_{0_{FSM}}, x_{dp_{FSM}}, X_{M_{FSM}})$ with $R_{FSM} = \emptyset$

$S_{F_a} = (T_{F_a}, R_{F_a})$ = a socket

$S_{F_b} = (T_{F_b}, R_{F_b})$ = b socket

Filters sit on pipes that connect the sockets of communicating modules. Pipes are defined by $P = (S_{P_a}, S_{P_b}, F_P)$, where

$S_{P_a} = (T_{P_a}, R_{P_a})$ = a socket

$S_{P_b} = (T_{P_b}, R_{P_b})$ = b socket

$F_P = (FSM, S_{F_a}, S_{F_b})$ = filter on pipe

Lastly, a system of modules is defined by $S = (M_S, F_S, P_S)$, where

$M_S = \{M_1, \dots, M_{n_M}\}$ = finite set of modules in the system

$F_S = \{F_1, \dots, F_{n_F}\}$ = finite set of filters in the system

$P_S = \{P_1, \dots, P_{n_P}\}$ = finite set of pipes connecting modules and filters

Although the MFSM architecture uses finite state machines in the modules, the module-based structure shows potential for individual modules to house different types of controllers depending on the specific system represented.

2.3 VERIFICATION AND VALIDATION

In order to test the robustness and correctness of the system controlled by the modMFSM, the system must be validated and verified. Validation proves that the system has the desired abilities, and verification proves that the system meets the required specifications and does not violate safety constraints (Zheng, Julien, Kim, & Khurshid, 2017). DES can be verified and validated through "test and simulation, model checking, and theorem proving" (Allen, Goh, & Tilbury, 2012). For verification and validation of the modMFSM structure, compositional verification, software testing, and model checking are considered. These methods complement each other for full testing of the system as accuracy of the methods separately are limited by the chosen scenarios, and the defined specifications, respectively (Lipka, Paška, & Potužák, 2014).

2.3.1 Compositional Verification

The compositional verification algorithm for MFSM theory checks that the controller will not send any module into a "dump" state, proving that there are no compositional errors, conflicted responses are avoided, and each module satisfies its corresponding filters (Endsley, 2004).

2.3.2 Simulation Testing

Software testing is a method for validating that the system meets the ability requirements and works as desired. Definition of the specific scenarios for simulation "from analytical description is widely recognized in literature" (Lipka et al., 2014). Unlike model checking, which relies on searching the states of the discrete event system, simulation testing is less susceptible to scaling issues for large systems. Scenario-based evaluation also allows assessment of the system's ability to handle situations that may not

be able to be tested through model checking (e.g. model checking does not force collisions to test a collision detection algorithm).

2.3.3 Model Checking

Model checking is used to prove adherence to desired system behavior. More specifically, model checkers search for violation of the specifications by providing counter-examples (Cassandras & Lafortune, 2011). To mitigate computational complexity of checking large systems, Hill et al. (R. C. Hill, Cury, de Queiroz, Tilbury, & Lafortune, 2010) show that "controllability and nonblocking of global systems [can be proven] through local checking" in hierarchical, modular systems without flattening the modules into a giant state machine, provided the system structure meets certain requirements, which are fully defined in (Richard Charles Hill, 2008). Additionally, through an evaluation of software testing versus software model checking, Beyer and Lemberger show that model checking can also efficiently identify bugs in the system (Beyer & Lemberger, 2017).

Desired system behavior can be defined using formal specifications. Three main categories of formal specifications are:

- Safety Specifications – to avoid illegal behavior.
- Invariant Specifications – to meet requirements specific to the system state.
- Liveness Specifications – to ensure absence of infinite loops, or to guarantee goal realization (Topcu, 2017).

Although system specifications commonly fall in these three categories, there exist formal specifications outside these categories.

Linear Temporal Logic (LTL) is a formal language that uses Boolean atomic propositions to define formal specifications such that discrete event systems can be

checked for fulfillment of desired system behavior. Software tools such as SMV, NuSMV, and SPIN for checking systems against LTL specifications are well-developed and widely used (Cassandras & Lafortune, 2011).

Although formal verification and validation is a vital step in proving the robustness and correctness of the model and software, it is beyond the scope of this paper, but will be the next step in future research.

2.4 Literature Review Summary

Supervisory controllers for large, complex systems like drilling rigs and snubbing units require guaranteed safe coordination of equipment and processes. Supervisory control efforts in the oil and gas industry have made significant progress, but there is still a need for supervisory controllers that can provide modularity, and functional scalability.

Supervisory controllers of event-driven discrete event systems are commonly modeled as FSMs, Petri nets, and MDPs. Choosing a model approach depends on the needs of the application. For large, complex systems like drilling rigs, it is beneficial to choose a model that can be decomposed into sub-systems, and maximizes decidability. Therefore, the MFSM architecture—which models large systems as a network of connected modules, each housing a subsystem modeled by trigger / response finite state machines—was used as the foundational structure on top of which modifications were made to meet the requirements for control of complex systems.

Chapter 3: modMFSM — An Extended Structure of Modular Finite State Machine (MFSM) System

Based on a reading of the literature and best industry practices, this research extends the modular finite state machine (MFSM) structure developed for reconfigurable machinery in the manufacturing industry, described in the previous chapter. The extended controller structure uses various models within modules, including guarded trigger / response finite state machines, in order to model a hierarchical supervisory control structure. The developed supervisory control structure (modMFSM) adapts to remote and on-site manual interventions, new equipment, collisions, and missing equipment. This section details the modular, supervisory controller architecture design.

3.1 Modification to TR FSMs in MFSM Structure

Endsley's MFSM structure uses trigger / response finite state machines (TR FSMs) in the modules and filters. In the proposed modified structure, transitions are triggered by events and/or fulfilled guards, or conditions, to respond to changes in system and environment variables to ensure safety and procedural compliance. The use of guards for input/output state machines is not a novel concept as shown in (Y. L. Chen & Lin, 2000), and have been applied to MFSMs through implementation of MFSMs as Event-Condition-Action (ECA) systems (E. E. Almeida et al., 2007; E. T. Almeida, Luntz, & Tilbury, 2005). In ECA systems, a rule affects transitions such that when an event happens, if the condition is satisfied, then the actions are performed. Although this implementation allows for the application of conditions for transitions based on observations of the environment, the MFSMs as ECA systems restricts the structure of the MFSMs such that there must be a main module to which peripheral modules are attached. The peripheral modules are only attached to the main module, and therefore,

cannot communicate with each other. If this ECA system implementation were to be applied to drilling automation, the main module would control the drilling procedure, and the peripheral modules would represent the sub-systems and equipment. However, for drilling automation, this ECA system structure does not suffice as it does not allow event detection modules, such as collision detection, to interact with equipment in parallel to the main module that controls the drilling process. Therefore, to relax the structural requirement of ECA systems, yet implement conditional transitions for MFSMs, guard conditions are captured by modifying the finite state machine transition definition.

$$\tau = \left(t_\tau, c_\tau, x_{s_\tau}, x_{d_\tau}, r_\tau\right)$$

where

t_τ = trigger for transition

c_τ = transition condition (e.g. $n_{drilled} < n_{desired}$)

x_{s_τ} = source state of the transition

x_{d_τ} = destination state of the transition

$r_\tau = \{r_{\tau,1}, \dots, r_{\tau,m_r}\}$ = sequence of responses as a result of the transition

3.2 System Variables

To allow for interaction of the supervisory controller with the environment for the guard conditions mentioned in the previous section, system variables are introduced to the MFSM architecture. These system variables are stored in a global database that can be accessed by all modules and filters in the system. The concept of using system

variables in conjunction with discrete event systems, including finite state machines and Petri nets, have been implemented previously in (Y. L. Chen & Lin, 2000; Gaudin & Deussen, 2007; Lennartson et al., 2014; Yang & Gohari, 2005). A particularly relevant piece of work is the incorporation of system variables with extended finite state machines in (Teixeira, Malik, Cury, & Queiroz, 2015). Incorporation of system variables in the modMFSM architecture allows modification of the system variables to be a responsive action in transitions. Therefore, the final definition of the transitions in the finite state machines is:

$$\tau = \left(t_\tau, c_\tau, x_{s_\tau}, x_{d_\tau}, r_\tau, r_{sys}\right)$$

where

t_τ = trigger for transition

c_τ = transition condition (e.g. $n_{drilled} < n_{desired}$)

x_{s_τ} = source state of the transition

x_{d_τ} = destination state of the transition

$r_\tau = \{r_{\tau,1}, \dots, r_{\tau,m_r}\}$ = set of responses as a result of the transition

$r_{sys} = \{r_{sys,1}, \dots, r_{sys,m_{sys}}\}$ = set of modifications to system variables as a result of the transition

3.3 Multilevel Hierarchy and Module Types

In the modMFSM controller structure, the modules are organized into eight categories for better understanding of each module's role in the system, and to allow for easy categorization of modules into the appropriate levels for hierarchical control, which

is naturally supported by the modularity of the modMFSM structure. Hierarchical supervisory control has been explored for various discrete event systems, including for MFSM systems (R. C. Hill et al., 2010; Richard Charles Hill, 2008; Ngo & Seow, 2014). The modularity of the MFSM structure allows the controller hierarchy to be easily rearranged to fit the desired system. This hierarchical structure aids in reducing complexity of the subsystem controllers, and allows for more flexible controller and equipment modularity.

For systems similar to drilling rigs and snubbing units, four levels are considered:

- Top Level – This level makes high-level decisions, and contains the *Operator GUI Module*, the *System Coordinator Module*, and the *Meta Module*.
- Middle Level – This level coordinates sub-systems to reach operational goals, and contains the *Operation Modules*. The *Operation Modules* represent the operational modes of the system.
- Low Level – This level of modules control the sub-systems directly, and contain the *Equipment Controller Modules*, *Manual Modules*, and *Physics-Based / Algorithm Modules*.
- Sub Level – This is the lowest level of the hierarchy, and contains the *Equipment Modules*, which interact directly with the machinery.

3.3.1 Operator GUI Module

The *Operator GUI Module* represents the GUI interface with which the human operator interacts. The most basic function of the *Operator GUI Module* is to activate the system. There is only one *Operator GUI Module* per system.

3.3.2 System Coordinator Module

The *System Coordinator Module* coordinates the operational goals for the system, and triggers the relevant *Operational Module*.

3.3.3 Meta Module

The *Meta Module* runs through detection of missing equipment and controllers by checking for disconnected sockets. If there is missing equipment or controllers, the *Meta Module* will automatically synthesize *Manual Modules*, and corresponding filters, sockets, and pipes to represent manual operator control of the system as replacements for missing equipment. This automatic synthesis procedure is in the simplest state, such that it allows multiple triggers, but only one response. Future research will be needed to develop an automatic synthesis algorithm that allows for multiple responses. Synthesis of discrete event systems controllers satisfying formally defined desired system behaviors is an ongoing and well established area of supervisory control research (Asarin, Maler, & Pnueli, 1994; Leduc, Dai, & Song, 2009; Malik & Teixeira, 2016; Thistle, 1996). However, the automatic generation of filters that correctly regulate interaction among modules has not been developed, but would alleviate the design effort needed and allow for more efficient controller reconfiguration. In his work, Hill has further investigated the construction of inter-module filters that satisfy global system properties; however, this algorithm requires designer input, but shows potential for implementation in a more sophisticated, automatic filter generation algorithm (Richard Charles Hill, 2008).

3.3.4 Operation Modules

The *Operation Modules* coordinate the activities of the sub-systems to reach the operational goal of the system as specified by the *System Coordinator Module*. Each *Operation Module* represents an operational mode of the system. For instance, an

automated drilling rig would have a tripping operational mode and a drilling operational mode, which would coordinate the running of pipe in and out of a well, and coordinate drilling the well, respectively. Each of these operational modes would be represented by a separate operation module to allow for isolated modification of operational procedures and sub-system involvement.

3.3.5 Equipment Controller Modules

These modules represent the supervisory controllers for sub-systems and coordinate the equipment activities.

3.3.6 Equipment Modules

This type of module models each piece of equipment, and send commands to the machinery.

3.3.7 Manual Module

The *Manual Module* encompasses a simple TR FSM that facilitates the transition of operations control between the supervisory controller and the human operator. The *Manual Module* is designed to be a versatile substitute for modules when a common error occurs that requires human intervention, or when operation modules, equipment controller modules, or equipment modules are missing. For instance, when the system identifies a piece of missing equipment, it creates a *Manual Module* that accepts triggers and produces a response as previously done by the missing equipment. Defined below is the framework for the TR FSM of a generic Manual Module that takes in one trigger and produces one response:

$$F_M = (X_{F_M},\ T_{F_M},\ R_{F_M},\ \tau_{F_M}, x_{0_{F_M}},\ x_{dp_{F_M}},\ X_{M_{F_M}})$$

where

X_{F_M} = {idle, manual_control}

T_{F_M}= {1.trigger, 2.auto}

R_{F_M} = {1.response, 2.manual}

τ_{F_M} = {(1.trigger, [ ], idle, manual_control, [2.manual]), (2.auto, [ ], manual_control, idle, [1.response])}

x_{0_M} = idle

x_{dp_M} = dump

X_{M_M} = idle

Defined below and shown in Figure 1 is the generic *Manual Module*:

$$M = (F_M, \{S_{M_1}, S_{M_2}, S_{M_3}, S_{M_4}\})$$

where

F_M = Described above.

S_{M_1} = {[1.trigger], [1.done]}

S_{M_2} = {[2.auto], [2.manual]}

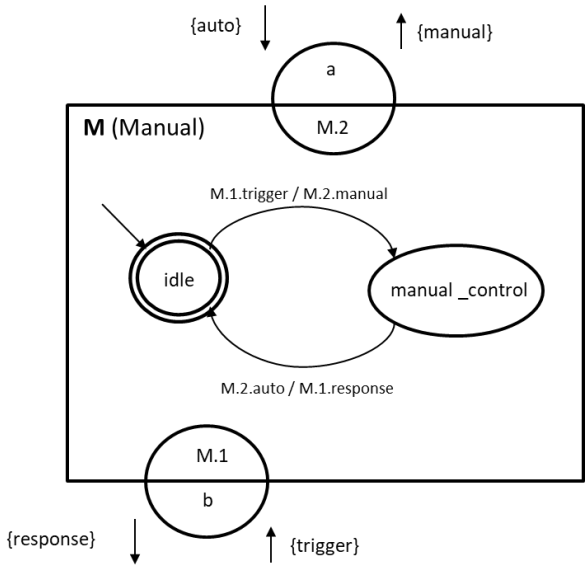

Figure 1. Generic *Manual Module* template.

3.3.8 Physics-Based / Algorithm Modules

A major advantage to using a module-based structure for building modular controllers is that the model type inside each module is not required to match the model types inside other modules as long as interaction requirements are met. For instance, one equipment module may encompass a trigger / response finite state machine while another equipment module hosts a Petri net. As long as the interior model is able to render the appropriate responses for the sockets of the module in order to interact with the other modules in the system, the choice of the interior model is only dependent on the purpose of the module.

This is especially useful if the supervisory controller requires physics-based algorithms for action selection based on environment observation, for instance in the case of error detection, or collision detection. In order to avoid detected collisions, the approach to resolution of events created by the collision detection module is to create a priority queue for these events that would be resolved first when executing the controller.

The use of such physics-based algorithm modules will be explored in the snubbing case study through the inclusion of collision detection for safe coordination of equipment, as further described in Chapter 4: Collision Detection. Although collision detection is the primary focus in this effort, the concept could be useful for other systems in the future. For example, on a drilling rig, close monitoring of cavings reveal important information about downhole conditions, which could trigger the need for a change in the operational conditions. A system is currently under development at the University of Texas at Austin to detect cavings at the mud shaker that utilizes a cuttings transport model (Han et al., 2017). This system could be included in a drilling rig modular supervisory controller as a module similar to a collision detection module.

Chapter 4: Collision Detection

To ensure safe operations, the supervisory controller must coordinate the subsystems such that the moving machinery never collide. Collision detection has been a fundamental concept in controls research, and an ongoing investigation topic in robotics, and computer graphics and simulations. The ideal collision detection system would run in the background and interrupt the system as necessary to avoid collision. This chapter discusses the collision detection approaches considered when designing the supervisory controller for the simplified case study system and for drilling.

In the proposed supervisory control framework, collision detection is implemented as a physics-based algorithm module, which interacts with the operation level module and the relevant equipment modules in the relevant subsystem. The collision detection module interacts with the other modules such that when a potential collision is detected, it will send commands to stop movement of equipment, store the current states of the equipment and operation modules, send commands to a collision resolution module for relocation of equipment to avoid collision, and return the equipment to the stored current state such that the original process can be resumed.

Out of the many developed collision detection algorithms, three general types of collision detection algorithms were explored when designing the supervisory controller for the simplified case study and for drilling:

- Heuristic Collision Detection
- Bounding Boxes Collision Detection
- B-Spline Surface Collision Detection

4.1 HEURISTIC COLLISION DETECTION

The heuristic approach simply restricts the movement of each piece of equipment dependent on the location of other active equipment. These relationships must be defined during the design of the model inside the collision detection module prior to implementation. For instance, in the case of snubbing operations, if the tongs are extended, then a minimum safe height is defined for the elevator, leaving a margin of distance before collision with the extended tongs, such that if the elevator reaches this minimum height, movement is stopped and the tongs moved out of the way before the elevator is allowed to resume descending. Although this is a relatively simple approach, it is not flexible to the addition or replacement of equipment. Bounding boxes is a more adaptive approach.

4.2 BOUNDING BOX COLLISION DETECTION

The widely used bounding box collision detection approach approximates the volume of each piece of equipment by considering an invisible box surrounding the equipment. If the bounding boxes of two objects overlap, then there is a possibility of collision. Common variations of the bounding box approximation are: axis-aligned bounding box (AABB), spheres, and oriented bounding box (OBB) (Kockara, Halic, Iqbal, Bayrak, & Rowe, 2007). Bounding box collision detection is another algorithm that can be included in the collision detection module for snubbing operations, chosen for its ability to adapt better to general shapes than other algorithms such as simplex-based or feature-based (Kockara et al., 2007).

For the case study in this report, ABB, where the sides of the rectangular bounding box align with the principal axis, which allow for accommodation of deforming objects, was implemented (Jiménez, Thomas, & Torras, 2001). Therefore, various

reference frames were defined to locate and orient the bounding boxes. A global reference frame was established at a central location. Each piece of equipment had a local reference frame attached to the center of the piece of equipment, and the equipment's bounding box was defined with respect to the local reference frame. All equipment movement was then calculated with respect to the global reference frame, which allowed detection of overlapping bounding boxes prior to the occurrence of a collision.

Since calibration of sensors is beyond the scope of the controller design this method is dependent on the assumption that the local reference frames of the equipment are properly calibrated with the origin of the global reference frame prior to commencement of operations.

4.3 B-Spline Surface Collision Detection

B-spline surfaces are able to approximate complex shapes more closely than polyhedral approximations. For B-spline approximations, it is possible to calculate minimum distance between B-spline surfaces for collision detection (Chang, Choi, Kim, & Wang, 2011). Although this type of collision detection provides more freedom of movement for the equipment and is less conservative, it requires knowledge of the specific geometry of the equipment. Exploration of this level of collision detection is outside the scope of this project.

4.4 Dynamic Object Discovery Collision Detection

Although the purpose of this supervisory controller is to track the location of the components of the controlled system, there can still exist components within the system workspace for which the locations are unknown. This occurs when unmodeled objects enter the workspace at uncertain times and frequencies. For instance, in a snubbing unit,

there remains a risk of human crew members entering the rig floor at uncertain points which could lead to collisions. Dynamically discovering objects that move in and out of the workspace can be performed via various sensors with possible combination with real-time image processing of cameras monitoring the environment. Dynamic object recognition and position detection has been extensively explored in robotics (S. Y. Chen, 2012). The idea of using image processing for drilling rig activity recognition is a new area of research in the oil and gas industry and can be leveraged for dynamic object discovery collision detection (Hegde, Awan, & Wiemers, 2018). Although this is a valuable path for exploration in future research efforts, this approach to collision detection calculation is beyond the scope of this effort.

Chapter 5: Case Study—Simple Conveyor Belt Drilling System

As a proof of concept, the modMFSM controller architecture was applied to a simple conveyor belt drilling system. The conveyor belt drilling system is formatted to demonstrate the controller's hierarchical structure, inclusion of physics-based models, modularity, and automatic generation of manual modules and corresponding filters. This section describes the simple conveyor belt drilling system, the design of its modular supervisory controller, and simulations of the implemented controller.

5.1 Definition of System and Environment

In order to design any controller, a model of the plant system and a thorough definition of the expected behaviors of the controlled system are necessary. This section defines the system components, process narrative, and assumptions needed to design a supervisory controller for a simple conveyor belt drilling system.

5.1.1 System Components

Identification of the hardware needed for automation of all activities is a critical step for designing a controller for coordination of equipment and processes. The conveyor belt drilling system contains two pieces of equipment: a drill robot, and a conveyor belt. Figure 2 presents a diagram of the conveyor belt drilling system.

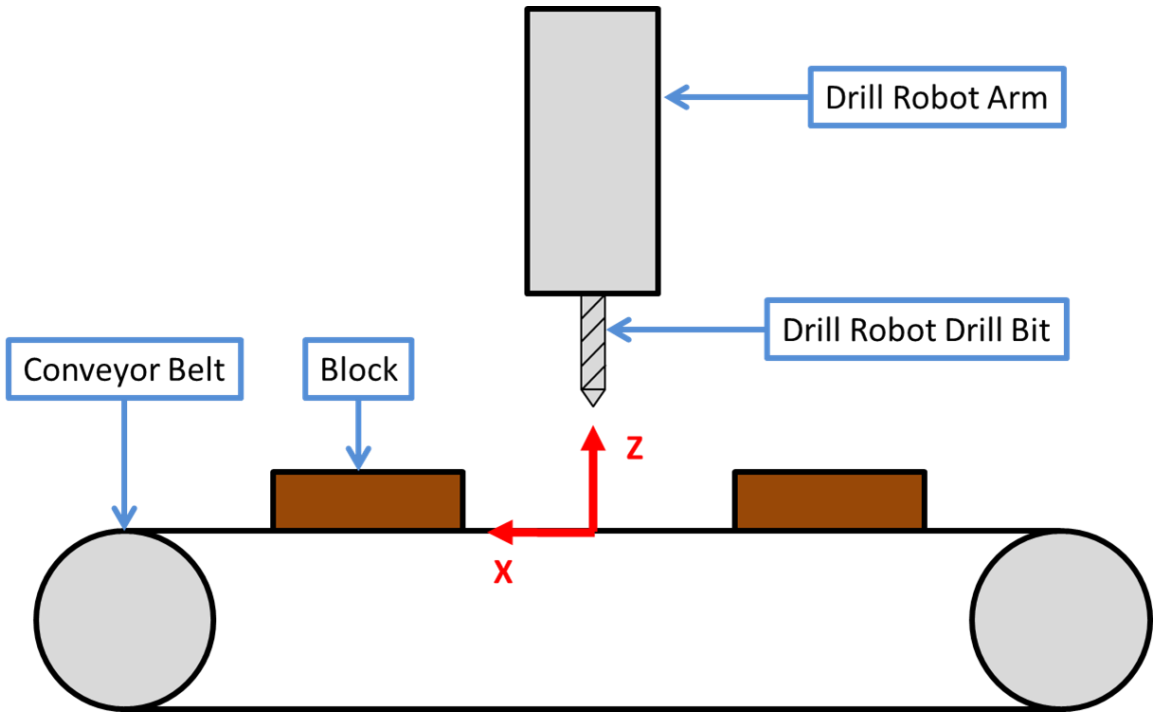

Figure 2. Diagram of the conveyor belt drilling system as used in the case study. The global reference frame is shown in red in the diagram with the y-axis coming out of the page.

Positions of the equipment and blocks to be drilled are relative to a global reference frame with an origin set at the base of the drill robot with the positive x-axis extending to the right of the robot, the positive y-axis extending through the front of the robot, and the positive z-axis extending up through the top of the robot. The drill arm and drill bit portion of the drill robot extends 0.1 m in the positive y-direction away from the base. The drill has a fixed x- and y- position, but can move up and down in the z-direction at 0.025 m/s. Additionally, when turned on, the drill bit rotates at 250 RPM about the z-axis. The conveyor belt lies 0.1 m in the positive y-direction, parallel to the x-axis. The conveyor moves blocks at 0.025 m/s in the negative x-direction.

In order to fully automate a system, appropriate sensors are crucial to provide controllers with feedback from the system and environment in order to best respond to events. The following sensors exist in the conveyor belt drilling system:

- Position sensor for the drill (positions of the bit can be calculated from the drill position)
- Linear velocity sensor for the drill
- Rotational velocity sensor for the drill bit
- Position sensor for the blocks
- Linear velocity sensor for the blocks

5.1.2 Process Narrative

A process narrative is the step-by-step action and decision-making procedures for particular operations. This documentation is essential for a thorough understanding of the work process in order to build a controller that guides the system through the correct flow of operations, while maintaining safety. Figure 3 shows the process narrative as a flowchart for drilling a block using the conveyor belt drilling system. In this flowchart, the yellow boxes represent a deciding factor for the next action. Although a collision detection process is not shown on the flowchart, it runs in the background, interrupting the main process if a collision is detected.

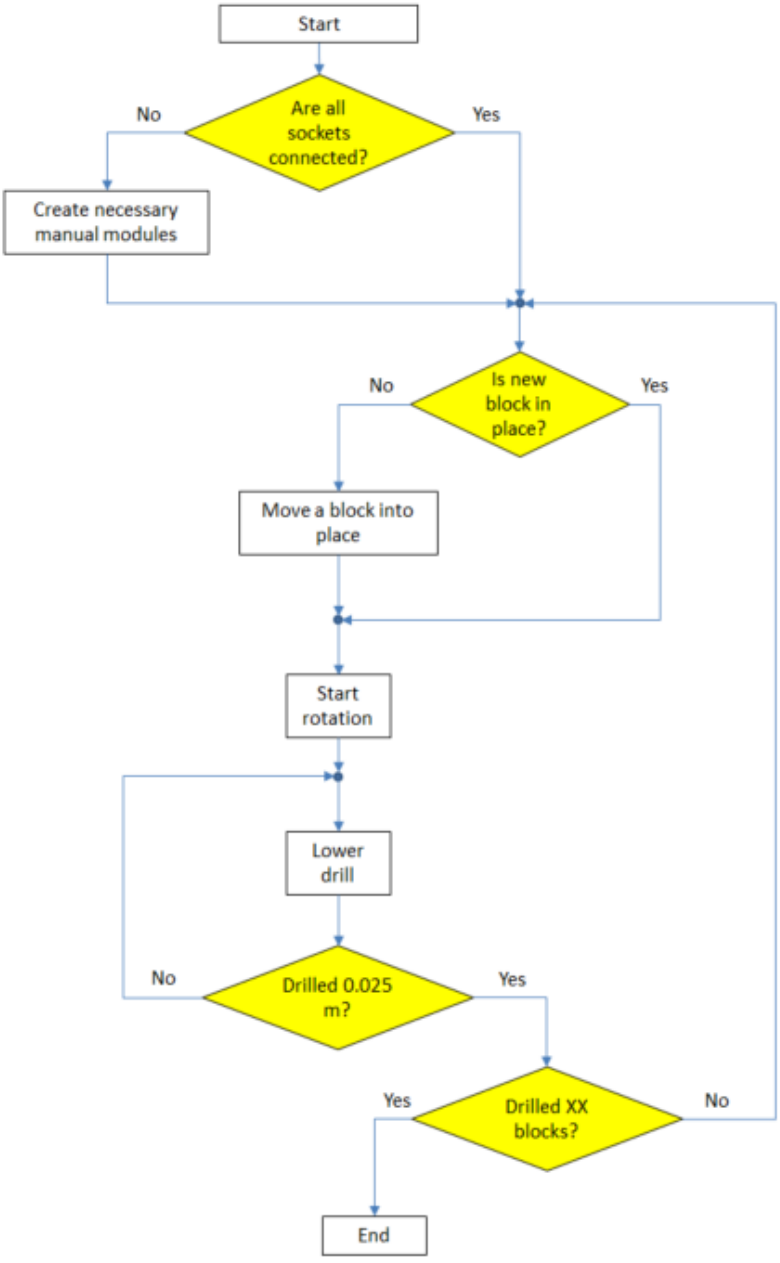

Figure 3. Process narrative flowchart for drilling a block in the simple conveyor belt drilling system.

5.1.3 Assumptions

Supervisory controllers work in highly dynamic and complex environments. When designing controllers, it may be necessary to make various assumptions about the system and environment in order to simplify the control structure to address events that are most likely to occur. Some of these assumptions may need to be addressed for improvement on the controller performance. However, work on resolving those assumptions is out of the scope of this project.

For the simple conveyor belt drilling system case study, the following are assumed:

- Global positions of each block are given by some sensor.
- The workspace is defined as $x = [-1.5\,m, 1.5\,m]$. Addition and removal of blocks from the workspace is assumed to be on an infinite conveyor belt such that when a block reaches $x = 1.5\,m$, it is considered to have left the workspace and a new block is added at $x = -1.5\,m$.
- Velocities of the drill robot and the conveyor belt are maintained at constant set points. Ramp up and ramp down times are negligible.
- Movement of the drill robot and the conveyor belt has no friction.
- No force or torque control is used for drilling.
- Sensors work in perfect condition without malfunctions or communication loss. Sensors are perfectly calibrated.
- Humans will not interrupt operations when the supervisory controller is in control.

5.2 CONVEYOR BELT DRILLING SYSTEM CONTROLLER

A 4-tier modMFSM controller was built for the conveyor belt drilling system. Since this system is only for feasibility proof of concept, a GUI was not built; therefore,

the *Operator GUI Module* was not used in the controller. Similarly, to minimize controller complexity for this simple system which only has one operational mode, the duties of the *System Coordinator Module* was incorporated into the *Meta Module* as shown in Figure 4. The controller includes two *Operation Modules* for the drilling process and collision resolution. There are two *Equipment Controller Modules*, the drill robot controller, and the conveyor belt controller, which command the drill robot and conveyor belt *Equipment Modules*, respectively. Collision detection for the drill robot and the blocks are included as the sole *Physics-Based / Algorithm Module* in the controller. The full design of the supervisory controller for the case study can be found in Appendix B – Case Study Controller Design.

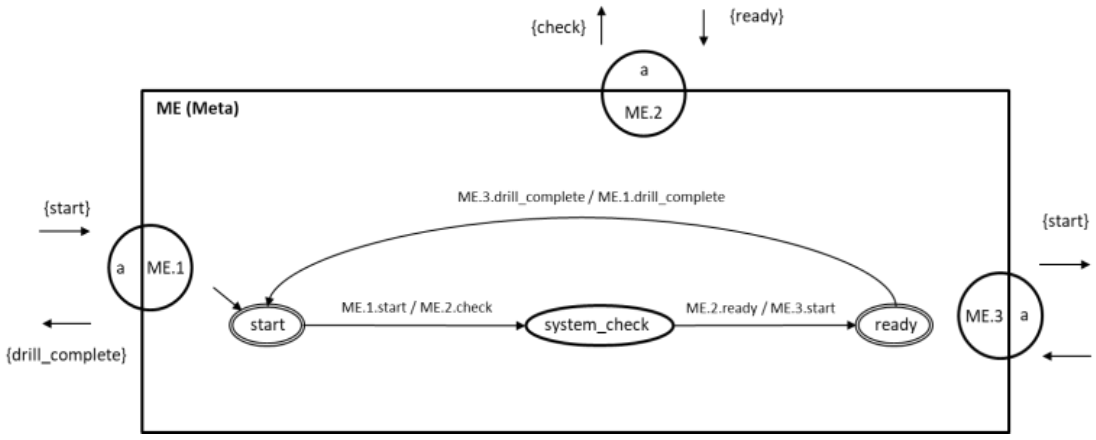

Figure 4. *Meta Module* for the conveyor belt drilling system.

5.3 COLLISION DETECTION AND RESOLUTION

The axis-aligned bounding box method was implemented for collision detection for the drill robot and the blocks in the workspace. The bounding boxes extended 0.025 m past the outermost parts of the components. The bounding box for the drill robot was split into two boxes: one for the drill arm, and one for the drill bit. The drill bit bounding

box was activated only when the conveyor belt moving to allow the drill bit bounding box to overlap with the bounding box of a block while raising and lowering the drill robot, and while drilling. Each block in the workspace had its own bounding box.

A collision was detected if the bounding boxes overlapped. The condition to be checked is as follows:

$$
\begin{aligned}
&(CB.fsm.xcur == 'moving' \\
&and\ (((any(2 * abs(X_B - X_{DR}) \leq BB_{BLOCK_X} + BB_{DR_X}) \\
&or\ any(2 * abs(X\_B\_DONE - X\_DR) \leq BB_{BLOCK_X} + BB_{DR_X})) \\
&and\ \left(2 * abs(Y_B - Y_{DR}) \leq BB_{BLOCK_Y} + BB_{DR_Y}\right) \\
&and\ (2 * abs(Z_B - Z_{DR_{CENTER}}) \leq BB_{BLOCK_Z} + BB_{DR_Z})) \\
&or\ (any(2 * abs(X_B - X_{BIT}) \leq BB_{BLOCK_X} + BB_{BIT_X}) \\
&and\ (2 * abs(Y_B - Y_{BIT}) \leq BB_{BLOCK_Y} + BB_{BIT_Y}) \\
&and\ (2 * abs\left(Z_B - Z_{BIT_{CENTER}}\right) \leq BB_{BLOCK_Z} + BB_{BIT_Z})))) \\
&or\ (CB.fsm.xcur != 'moving' \\
&and\ ((any(2 * abs(X_B - X_{DR}) \leq BB_{BLOCK_X} + BB_{DR_X}) \\
&or\ any(2 * abs\left(X_{B_{DONE}} - X_{DR}\right) \leq BB_{BLOCK_X} + BB_{DR_X})) \\
&and\ (2 * abs(Y_B - Y_{DR}) \leq BB_{BLOCK_Y} + BB_{DR_Y}) \\
&and\ (2 * abs\left(Z_B - Z_{DR_{CENTER}}\right) \leq BB_{BLOCK_Z} + BB_{DR_Z}))
\end{aligned}
$$

where

$CB.fsm.xcur$ = current state of the Conveyor Belt (CB) Finite State Machine (FSM)

X_B, Y_B, Z_B = x, y, and z positions of the blocks

$X_{B_{DONE}}$ = x position of the blocks that have been drilled

$X_{BIT}, Y_{BIT}, Z_{BIT}$ = x, y, and z positions of the drill bit

$BB_{BLOCK_X}, BB_{BLOCK_Y}, BB_{BLOCK_Z}$ = length of the bounding box of each block in the x, y, and z directions

$BB_{DR_X}, BB_{DR_Y}, BB_{DR_Z}$ = length of the bounding box of the drill robot in the x, y, and z directions

$Z_{DR_{CENTER}}$ = z position of the center of the drill robot

$Z_{BIT_{CENTER}}$ = z position of the center of the drill bit

5.4 IMPLEMENTATION

The conveyor belt drilling system and controller were implemented in Python 3.6.5 from Anaconda, Inc. Simulations as described in Section 5.5 Simulations were run on an Intel® Core™ i7-7500U (2.70GHz) processor.

The modMFSM events (triggers and responses), finite state machines, sockets, modules, filters, pipes, and system of modules were implemented using Python classes. The *event* class was created to define each trigger and response event. Each *event* object contains the following attributes:

- Name of the *event* object, stored as a string
- Socket through which the event enters or exits, stored as a *socket* object
- Event name without the socket prefix, stored as a string
- Finite state machine contained by the module in which the event occurs , stored as an *fsm* object

The *fsm* class defines the finite state machines for the modules and filters. The attributes of each *fsm* object are listed below:

- Name of the *fsm* object, stored as a string
- Finite set of states, stored as a list of strings
- Finite set of trigger events, stored as a list of *event* objects
- Finite set of response events, stored as a list of *event* objects
- Finite set of transitions, stored as a tuple of tuples, in which each tuple represents a transition and contains the trigger *event* object, transition condition string, source state string, destination state string, set of response *event* objects, and set of system variable changes stored as a tuple of strings
- Initial state, stored as a string
- Dump state, stored as a string
- Marked states, stored as a list of strings
- Current state, stored as a string
- Saved state, stored as a string

The *socket* class represents the sockets found on the modules. Each *socket* object represents each individual socket with the following attributes:

- Name of the *socket* object, stored as a string
- Finite set of trigger names without the socket prefix, stored as a list of strings
- Finite set of response names without the socket prefix, stored as a list of strings

The modMFSM modules are represented by the *module* class, for which each object has the following attributes:

- Name of the *module* object, stored as a string
- Finite state machine contained in the module, stored as an *fsm* object
- Finite set of sockets in the module, stored as a list of *socket* objects

The *filters* class, which represents the modMFSM filters, is similar in structure to the *module* class. Each *filters* object has the following attributes:

- Name of the *filters* object, stored as a string
- Finite state machine contained in the filter, stored as an *fsm* object
- Socket A, stored as a *socket* object
- Socket B, stored as a *socket* object

The *pipe* class defines the modMFSM pipe which connects modules. Each *pipe* object has the below attributes:

- Name of the *pipe* object, stored as a string
- Socket A, stored as a socket object
- Socket B, stored as a socket object
- Filter on the *pipe* object, stored as a *filters* object

The last class created for implementation of the modMFSM structure was the *systemOfModules* class which defines the system of modules, filters, and pipes that make up the supervisory controller. Each *systemOfModules* object includes the following attributes:

- Name of the *systemOfModules* object, stored as a string
- Finite set of modules in the system, stored as a list of *module* objects
- Finite set of filters in the system, stored as a list of *filters* objects
- Finite set of pipes in the system, stored as a list of *pipe* objects

System variables were stored globally to allow access for all functions. Changes to the system variables as a result of transitions were stored in the definition of the transitions as a set of strings, e.g. the " CHECK_DONE = 'True' " string indicated that after the corresponding transition, the variable CHECK_DONE will be set to True. Since this action was stored as a string, execution of the action was done with Python's *exec* function. Similarly, the conditions for transitions were stored as strings and evaluated using Python's *eval* function. In addition to the system variables, two lists of *event*

objects were stored globally. These lists were the queues for the event triggers in the main system operations, and the event triggers for collision detection and collision resolution, respectively.

In order to allow the collision detection module to interrupt operations in the event of a possible collision, threading was used to implement two threads: one for the main operations, and one for the collision detection algorithm. This allowed the collision detection algorithm to run in the background to check for collisions, and interrupt the main thread as necessary. The algorithms in both threads are situated inside *while loops* that run while the Boolean CONTINUE variable is true. This allows the code to terminate the threads once a termination condition has been met. It is expected that if more *Physics-Based / Algorithm Modules* are added to the modMFSM controller, a new thread would be implemented for each physics-based module that requires interruption of the main operations.

The main operations thread runs the transitions of the system. Inside the *while loop* of the main operations thread, an *if* statement around the main operations code checks if the Boolean COLLISION variable is true, indicating whether there is a potential collision. The main operations algorithm only runs if COLLISION is false, so that if a possible collision exists, the controller will suspend motion in the system until the collision is resolved. If there is no collision, the main operations thread first calls an *update_system* function. This function updates the system variables for simulation purposes. (For implementation of a controller that interacts with a live system, the *update_system* function would not be necessary.) After the system variables are updated, the algorithm checks for transitions that have no trigger event but have been triggered by a change in system variables, and adds response events to the main operations queue. After resolving these transitions, the algorithm pops the next event off the main

operations event queue, searches the system for the transition triggered by the event, updates the relevant finite state machine state, adds response events to the main operations event queue, and updates relevant system variables as necessary. The algorithm then returns to the beginning of the *while loop* code.

The collision detection thread checks for potential collisions in the background while the main operations thread runs. In order to check for collisions using the most up-to-date positions and velocities of the equipment, the collision detection thread first calls the *update_system* function to update the system variables. Then, the code checks for collisions per the condition described in Section 5.3 Collision Detection and Resolution. If a collision is detected, the Boolean COLLISION variable is set to True, the current state of all modules are saved, motion is suspended, and the collision is resolved per the collision resolution module. Events from the collision detection and collision resolution algorithms are stored in a collision event queue to keep them separate from the events of the main operations. Once the collision has been resolved, the modules are returned to their saved states, the COLLISION variable is set to False, and main operations are resumed.

5.5 SIMULATIONS

Preliminary testing of the modular supervisory controller concept feasibility focused on simulation testing, the following three simulation cases were implemented:

- Error-Free Operations
- Forced Collision
- Missing Conveyor Belt Machine

Since a GUI was not created for the simplified case study system, the *Meta Module* initial state was set to be “system_check” and an initial event of ME_2_check

was added to the main operations event queue to simulate a GUI starting the system. Additionally, a time limit was added to the execution of the code to allow for termination in case of an infinite loop.

Prior to implementation of the modMFSM structure on a larger, more complex system for snubbing operations, compositional verification will need to be done, and LTL system requirements will need to be defined for formal verification via model checking. The expected transitions and output of the simulation code for the simulation cases can be found in Appendix C – Case Study Simulation Transitions and Code Output.

5.5.1 Case 1: Error-Free Operations

This simulation case runs the system through error-free operations of drilling three blocks. "Error-free" here means that the system is expected to follow the process narrative without collisions or malfunctioning machinery. The Case 1 simulation ran in 89.9 seconds.

5.5.2 Case 2: Forced Collision

This simulation case tests the ability of the collision detection module to identify a potential collision and interrupt the main operations to resolve the potential collision. The initial positions of the drill robot and blocks in the workspace were set such that the bounding boxes of the drill robot arm and a block overlap, forcing a potential collision to be detected. In Case 2, only one block was to be drilled. The Case 2 simulation ran in 21.2 seconds.

5.5.3 Case 3: Missing Conveyor Belt Machine

The third simulation case tests the ability of the controller to detect disconnected sockets of a piece of missing equipment and automatically generate manual modules for

replacement. To do so, the code for the conveyor belt related classes were commented out. For simplicity, manual intervention was simulated by a three second pause and manipulation of the system variables to give the appearance that a human had added or removed a block from the workspace. Three blocks were expected to be drilled in Case 3.

The simulation of Case 3 successfully detected the missing conveyor belt, and created two manual modules: one for the normal operations of the conveyor belt, and one for the collision detection commands for the conveyor belt. The simulation ran for 45.2 seconds.

5.6 Case Study Results Summary

The successful simulations of the three test cases show promise for feasibility of the modMFSM controller structure, but a full verification for compositional and formal correctness is necessary to complete the validation and verification process.

Some observations were of the controller implementation that must be considered when implementing on a larger, more complex system. Selection of set points for machinery velocity and bounding box margins are dependent on program run time. The time between the detection of a potential collision and the collision happening must not be less than the time needed for the process to halt the moving parts once a potential collision is detected. Additionally, a forced delay was needed in the main thread to allow the collision detection thread to run its check before the next action occurs. The duration of the delay can be optimized depending on the program run time and equipment machine limits.

Chapter 6: Extension of modMFSM to Snubbing Operations

This section describes the proposed approach for implementation of the modified modular supervisory controller to snubbing operations.

6.1 Definition of System and Environment

In order to design any controller, a model of the plant system and a thorough definition of the expected behaviors of the controlled system are necessary. Section 6.2 Process Narrative defines the process narrative, system components, system requirements, and environment assumptions for the case of designing a supervisory controller for snubbing operations. In addition to referencing available literature on snubbing like Grace (2003) and Prebeau-Menezes (2013), documentation of system and environment definitions was completed thanks to the permission of Superior Energy Services, Inc. to visit one of their mechanized snubbing standalone units. With permission from Superior Energy Services, Inc., information was gathered through:

- Video and photograph recordings of operations
- Manually recorded notes of operations and crew practices
- Interviews with the crew

6.2 Process Narrative

A process narrative is the step-by-step action and decision-making procedures for particular operations. This documentation is essential for a thorough understanding of the work process in order to build a controller that guides the system through the correct flow of operations, while maintaining safety.

For ease of understanding, the snubbing operations process narrative is represented in a series of flowcharts. Shown below is the flowchart created for the

snubbing out and stripping out process for a mechanized snubbing rig. In this flowchart, the yellow boxes represent a deciding factor for the next action. The blue boxes indicate that action requires a procedure represented by its own flowchart in the series, which shares the name as the action in the blue box. The full series of flowcharts for snubbing operations can be found in Appendix D – Process Narrative Flowcharts for Snubbing.

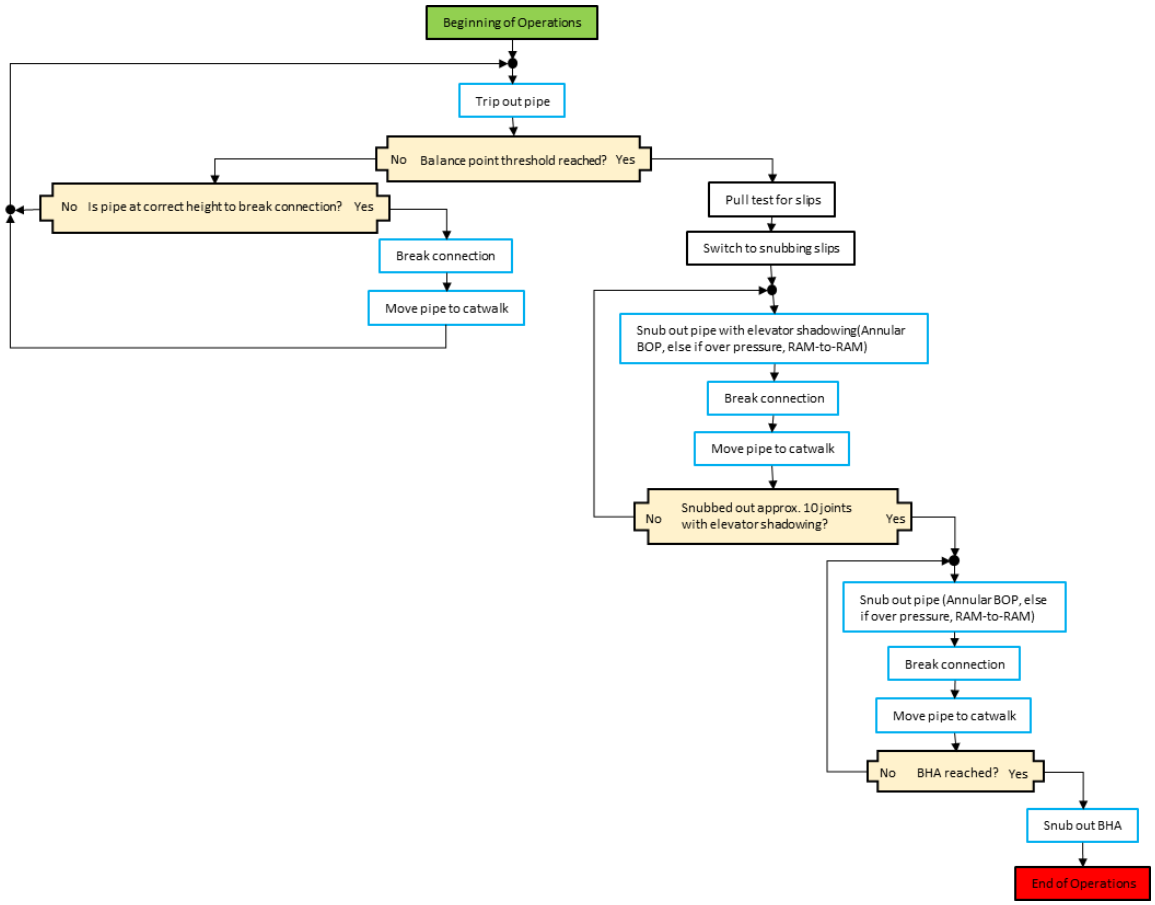

Figure 5. Process narrative flowchart for snubbing / stripping out pipe.

6.3 System Components

Documenting the process narrative also allows for identification of the hardware needed for automation of all activities, which is a critical step for designing a controller for coordination of equipment and processes.

Appendices

Appendix A – Snubbing Unit Equipment describes general equipment and machinery for snubbing units, which exist as both standalone and rig-assist units. For the purpose of this case study, a mechanized, standalone snubbing unit is considered. However, the proposed supervisory controller structure is designed to be adaptable to all types of rigs, accommodating for various equipment combinations.

In order to fully automate a system, appropriate sensors are crucial to provide controllers with feedback from the system and environment in order to best respond to events. Table 1 shows the sensors necessary to automate bringing new pipe to the well center for snubbing operations. From the table, it can be seen that at least twelve sensors are needed for autonomously bringing new pipe to the well center. A full table of sensors needed for automating snubbing can be found in Appendix E – Snubbing Automation Sensors. From left to right, the columns describe the variable observed, whether sensor hardware exists and its type, suggested sensor hardware if not already available, and the current sensing method for the variable.

Table 1. System variables and corresponding sensors for bringing new pipe to the well center.

Variable	**Sensor Hardware**	**Suggested Hardware**	**Current Method**
Stabbing Plate Alignment	None	Camera	Visual / Manual
Well Pressure	Casing Pressure Gauge	N/A	Sensor
Joint Counter	None	N/A (Build into controller)	Visual / Manual
Height of Pipe for Connection	Block Position	N/A	Sensor
Elevator Position			Visual / Manual
Elevator Closed on Pipe	Built-In Open/Close Indicator	N/A	Visual / Manual
Catwalk Height	Built-In Sensor (Possible Limit Switch)	N/A	Sensor
Elevator Height at Pipe	None	Camera	Visual / Manual
Catcher's Mitt (Extended/Retracted)	Built-In Indicator (Existence Uncertain)	N/A	Visual / Manual
Tongs (Extended/Retracted)	Built-In Indicator - Well center is set point	N/A	Sensor
Pipe in Catcher's Mitt	None	Contact or Proximity Sensor	Visual / Manual
Tongs Torque	Built-In Torque Sensor	N/A	Sensor
Pipe End in Catwalk	None	Catwalk Camera	Visual / Manual
Pipe in Skate Basket			Visual / Manual
Tongs Dies Failure	Built-In Torque Sensor; Motor rotation encoder on pipe	N/A	Sensor
Connection Made			Sensor / Visual

6.4 Formal Specification of System Requirements

As described in Section 2.3.3 Model Checking, an important step in the design process of a controller is to specify the desired behavior of the controlled system. Since linear temporal logic (LTL) specifications work well with finite state machines for model checking and contract-based design, required system behaviors are defined by LTL specifications. Table 2 displays LTL specifications for operations for bringing new pipe to the well center. This specific operational mode focuses primarily on safety specifications. A table of all formal specifications for snubbing operations can be found in Appendix F – Snubbing Automation Formal Specifications.

These formal specifications will also be defined through creation of contracts to specify the dependencies among modules to ensure safe operations. Formal specifications that are specific to a procedure or equipment modeled inside a module by a finite state machine will be used for model checking.

Table 2. Formal specifications for system requirements in operations for bringing new pipe to the well center.

Type	Requirement Behavior	Linear Temporal Logic
Safety	Snubbing Jack Height <= Maximum Safe Jack Height	**G** (Jack_Height < Max_Jack_Safe_Height)
Safety	Snubbing Jack Height >= Minimum Safe Jack Height	**G** (Jack_Height > Min_Jack_Safe_Height)
Safety	Do not close empty RAMs	**G** (RAM_empty **->** RAM_open)
Safety	Do not close slips on a tool or connection	**G** (((Jack_Height < Tool_Joint_Max) **^** (Jack_Height > Tool_Joint_Min)) **->** Slips_Open)
Safety	If catcher's mitt or tongs extended, elevator position must be above safe lower limit, and snubbing jack must be at the minimum safe jack height.	**G** ((Mitt_Extended **v** Tongs_Extended) **->** ((Elevator_Height >= Safe_Lower_Limit) **^** (Jack_Height = Min_Jack_Safe_Height)))
Safety	If elevator position below safe lower limit or snubbing jack above safe lower limit, catcher's mitt and tongs must be retracted.	**G** (((Elevator_Height < Safe_Lower_Limit) **v** (Jack_Height >Min_Jack_Safe_Height)) **->** (**!**Mitt_Extended **^ !**Tongs_Extended))
Safety	If snubbing jack not at safe lower limit, elevator position must be above safe lower limit, and catwalk cannot push up pipe.	**G** ((Jack_Height > Min_Jack_Safe_Height) **->** ((Elevator_Height > Safe_Lower_Limit) **^** (Skate_Basket < Max_Skate_Height)))
Safety	Elevator height should always remain more than a margin above the snubbing jack height.	**G** (Elevator_Height > Jack_Height + Margin)
Safety	If connection made or connection not broken, catcher's mitt must not be extended.	**G** ((Connection_Made **v !**Connection_Broken) **- > !**Mitt_Extended)

6.5 SYSTEM AND ENVIRONMENT ASSUMPTIONS

Supervisory controllers work in highly dynamic and complex environments. When designing controllers, it may be necessary to make various assumptions about the system and environment in order to simplify the control structure to address events that are most likely to occur. Some of these assumptions may need to be addressed for improvement on the controller performance. However, work on resolving those assumptions is out of the scope of this project.

For the snubbing case study, the following are assumed:

- Sensors will work in perfect condition without malfunctions or communication loss.
- Sensors are perfectly calibrated.
- Missing sensors for required variables (e.g. stabbing alignment) exist.
- Snubbing personnel are situationally aware. Should manual control be necessary at any time, the crew will respond promptly and appropriately.
- Snubbing personnel will not interrupt operations when the supervisory controller is in control.

As the case study is simulated, new assumptions and restrictions on equipment ability may need to be added in order to reduce controller complexity and computational effort.

6.6 Definition of Modular Controller Structure for Snubbing Automation

Since the proposed modMFSM structure encompasses controllers in modules that can represent system or equipment controllers, the MFSM structure naturally allows for hierarchical control. The modularity of the MFSM structure allows the controller hierarchy to be easily rearranged to fit the desired system. This hierarchical structure aids in reducing complexity of the subsystem controllers, and allows for more flexible controller and equipment modularity.

As shown in Figure 6, the snubbing operations modMFSM supervisory controller modules are comprised of the same module types as used for the simplified conveyor belt drilling system controller, albeit with the addition of the *Operator GUI* Module, and the *System Coordinator Module.* For snubbing operations, control is split into four primary levels as described in Section 3.3 Multilevel Hierarchy and Module Types:

- Top Level

- Middle Level
- Low Level
- Sub Level

These levels are reviewed in the next subsections with examples from the snubbing controller.

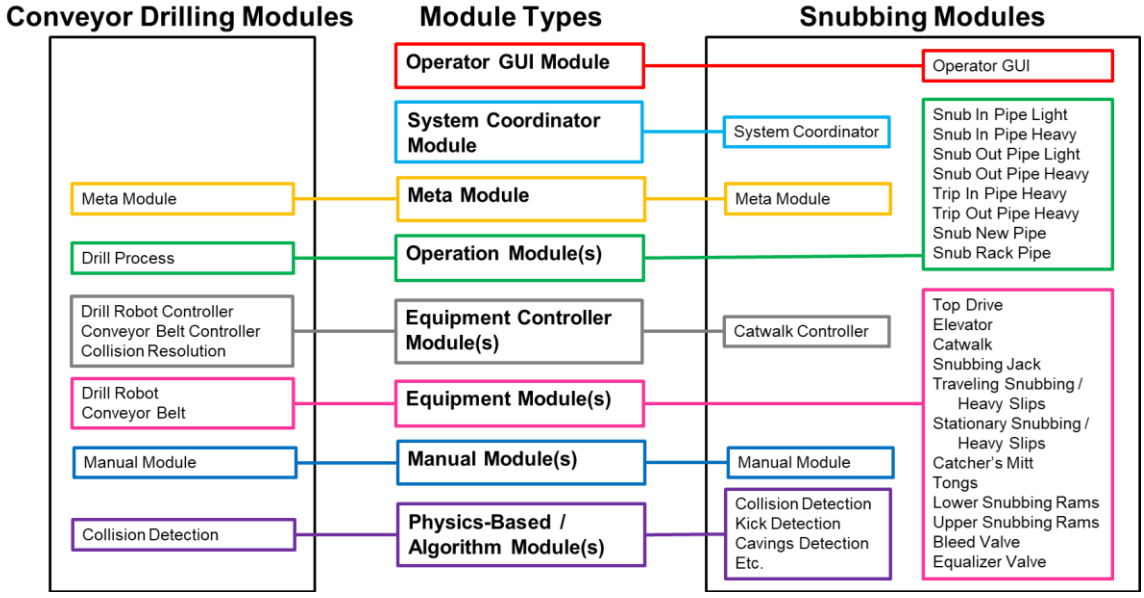

Figure 6. Comparison of the modules for the conveyor belt drilling system with those for snubbing operations.

6.6.1 Top Level

The Top Level of control is the highest level of control in the structure and consists of the *Operator GUI Module*, the *System Coordinator Module* that selects the mode of operation in which the system is currently operation (e.g. Snubbing In Pipe Heavy (*SIPH*), Snubbing New Pipe (*SNP*), etc.), and the *Meta Module*. Figure 7 shows how the *System Coordinator* module interacts with the *Meta Module* and modules from the Middle Level of control.

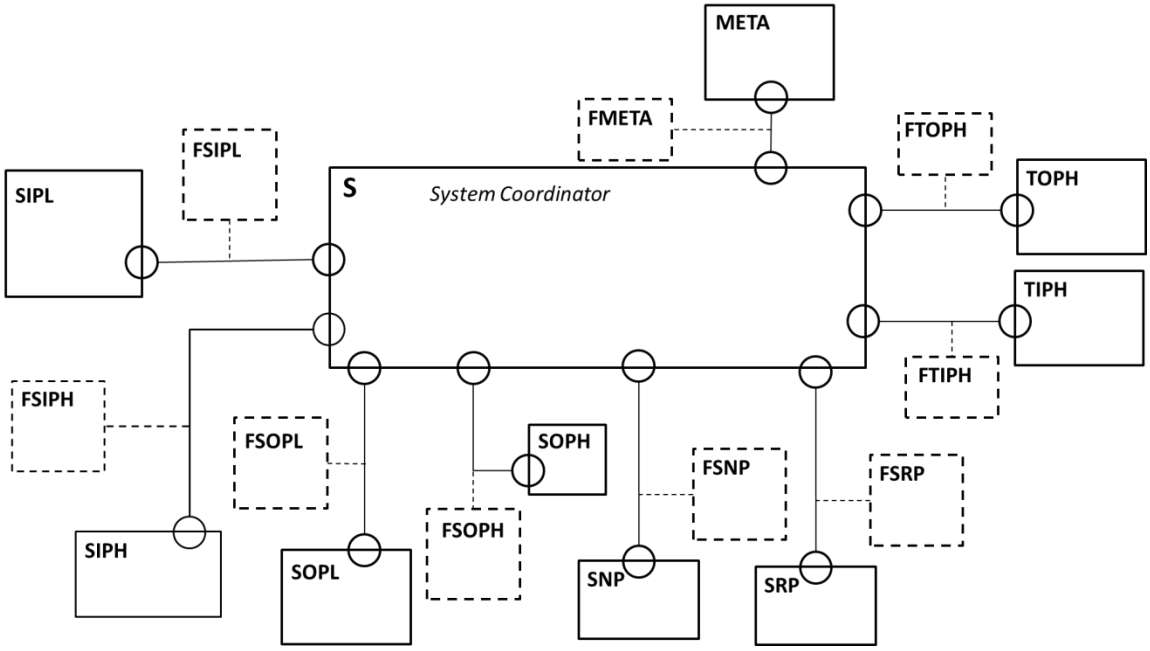

Figure 7. High-level diagram of interaction between the *System Coordinator Module*, *Meta Module*, and *Operation Modules*.

6.6.2 Middle Level

Like many complex operations, snubbing operations can be divided into sub-operations, which will be referred to as operation modes, each of which follows different procedures and requires a different set of equipment. Operation modes are independent of each other; therefore, only one operation mode is active at a time, which simplifies the controller by reducing the tracked states to those associated with the processes and equipment of the active operation mode. The Middle Level of control consists of the modules which control the procedures of each operation mode, sending commands to the relevant equipment controllers in its subsystem and reacting to subsystem and environment events. The following is the definition of the guarded, trigger / response

finite state machine for the Snubbing New Pipe Operation Module (*SNP*), which controls bringing new pipe from the racked position to the well center as shown in Figure 8:

$$F_{SNP} = (X_{F_{SNP}}, T_{F_{SNP}}, R_{F_{SNP}}, \tau_{F_{SNP}}, x_{0_{F_{SNP}}}, x_{dp_{F_{SNP}}}, X_{M_{F_{SNP}}})$$

where

$X_{F_{SNP}}$ = {inactive, prep_catchers_mitt, prep_tongs, prep_bails, prep_elevator, new_pipe, grab_pipe, extend_catchers_mitt, extend_tongs, lift_pipe, pipe_to_catchers_mitt, error, pipe_to_conn_height, check_alignment, stab_pipe, connection}

$T_{F_{SNP}}$= {1.new_pipe, 2.extend_done, 2.retract_done, 3.retract_done, 3.extend_done, 3.open_done, 3.close_done, 3.stop_done, 4.new_pipe_ready, 5.extend_done, 5.retract_done, 5.make_connection_done, 6.error_done, 6.fix_alignment}

$R_{F_{SNP}}$ = {1.new_pipe_ready, 2.extend, 2.retract, 3.retract, 3.extend, 3.open, 3.close, 3.raise, 3.lower, 3.stop, 4.new_pipe, 5.extend, 5.retract, 5.make_connection, 6.error_done, 6.align_fixed}

$\tau_{F_{SNP}}$ = {(1.new_pipe, [ ], inactive, prep_catchers_mitt, [2.retract], [ ]),
(2.retract_done, [ ], prep_catchers_mitt, prep_tongs, [5.retract] , [ ]),
(5.retract_done, [ ], prep_tongs, prep_bails, [3.retract] , [ ]),
(3.retract_done, [ ], prep_bails, prep_elevator, [3.open] , [ ]),
(3.open_done, H_elev==H_elev_pipe, prep_elevator, new_pipe, [3.stop] , [ ]),
([ ], H_elev==H_elev_pipe, prep_elevator, new_pipe, [3.stop] , [ ]),
(3.open_done, H_elev<H_elev_pipe, prep_elevator, prep_elevator, [3.raise] , [ ]),
(3.open_done, H_elev>H_elev_pipe, prep_elevator, prep_elevator, [3.lower] , [ ]),
(3.stop_done, [ ], new_pipe, new_pipe, [4.new_pipe] , [ ]),
(4.new_pipe_ready, [ ], new_pipe, grab_pipe, [3.extend] , [ ]),
(3.extend_done, [ ], grab_pipe, grab_pipe, [3.close] , [ ]),
(3.close_done, [ ], grab_pipe, extend_catchers_mitt, [2.extend] , [ ]),
(2.extend_done, [ ], extend_catchers_mitt, extend_tongs, [5.extend] , [ ]),
(5.extend_done, [ ], extend_tongs, lift_pipe, [3.raise] , [ ]),
([ ], H_elev == H_elev_clear_CW, lift_pipe, lift_pipe, [3.stop] , [ ]),
(3.stop_done, [ ], lift_pipe, lift_pipe, [3.retract] , [ ]),
(3.retract_done, [ ], lift_pipe, pipe_to_catchers_mitt, [ ] , [ ]),

([ ], pipe_catchers_mitt==false, pipe_to_catchers_mitt, error, [6.error] , [ ]),
(6.error_done, error, pipe_to_catchers_mitt, [ ] , [ ]),
([ ], pipe_catchers_mitt==true, pipe_to_catchers_mitt, pipe_to_catchers_mitt, [2.retract] , [ ]),
(2.retract_done, pipe_to_catchers_mitt, check_alignment, [ ] , [ ]),
([ ], aligned==false, check_alignment, check_alignment, [6.fix_alignment] , [ ]),
(6.align_fixed, check_alignment, check_alignment, [ ] , [ ]),
([ ], aligned == true && H_elev>H_elev_conn, check_alignment, pipe_to_conn_height, [3.lower] , [ ]),
([ ], aligned == true && H_elev<H_elev_conn, check_alignment, pipe_to_conn_height, [3.raise] , [ ]),
([ ], aligned == true && H_elev==H_elev_conn, check_alignment, pipe_to_conn_height, [3.stop] , [ ]),
([ ], H_elev<H_elev_conn, pipe_to_conn_height, pipe_to_conn_height, [3.raise] , [ ]),
([ ], stabbed==true && H_elev==H_elev_conn,pipe_to_conn_height, connection, [3.stop] , [ ]),
(3.stop_done, [ ], connection, connection, [5.make_connection] , [ ]),
(5.connect_done, [ ], connection, inactive, [1.new_pipe_ready] , [ ])}

$x_{0_{F_{SNP}}}$ = inactive

$x_{dp_{F_{SNP}}}$ = dump

$X_{M_{F_{SNP}}}$ = inactive

Defined below is the *Operation Module* for Snubbing New Pipe:

$$SNP = (F_{SNP}, \{S_{F_{SNP_1}}, S_{F_{SNP_2}}, S_{F_{SNP_3}}, S_{F_{SNP_4}}, S_{F_{SNP_5}}, S_{F_{SNP_6}}\})$$

where

F_{SNP} = Described above.

$S_{F_{SNP_1}}$ = {[1.new_pipe], [1.new_pipe_ready]}

$S_{F_{SNP_2}}$ = {[2.extend_done, 2.retract_done], [2.extend, 2.retract]}

$S_{F_{SNP_3}}$ = {[3.retract_done, 3.extend_done, 3.open_done, 3.close_done, 3.stop_done], [3.retract, 3.extend, 3.open, 3.close, 3.raise, 3.lower, 3.stop]}

$S_{F_{SNP_4}}$ = {[4.new_pipe_ready], [4.new_pipe]}

$S_{F_{SNP_5}}$ = {[5.extend_done, 5.retract_done, 5.make_connection_done], [5.extend, 5.retract, 5.make_connection]}

$S_{F_{SNP_6}}$ = {[6.error_done, 6.fix_alignment], [6.error_done, 6.align_fixed]}

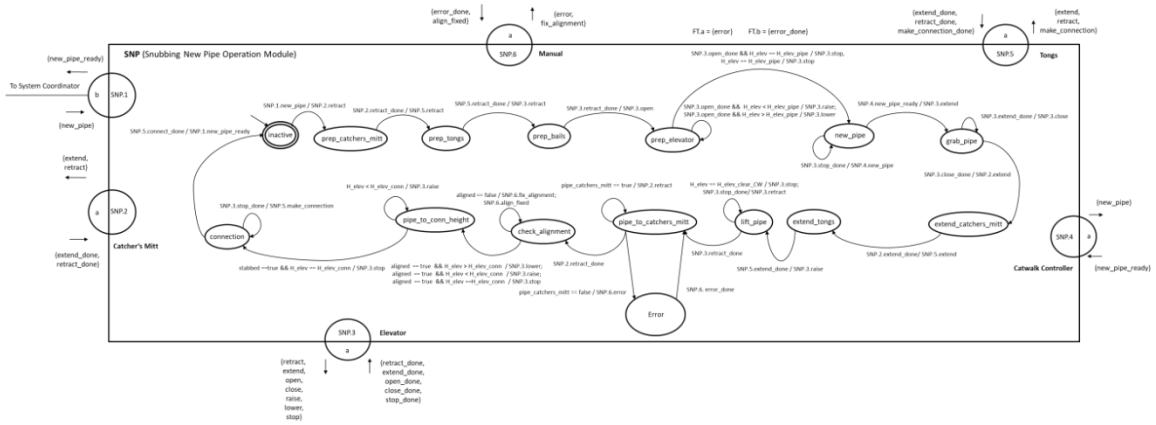

Figure 8. High-level view of schematic for the Snubbing New Pipe *Operation Module*. (See Appendix G – Snubbing Modular Controller Preliminary Designs for the full-size diagrams of the completed snubbing operation modules.)

6.6.3 Low Level

The Low Level of controls manages the communication between the *Operation Modules* and the sub-systems, which include *Equipment Controller Modules*, *Manual Modules*, and *Physics-Based / Algorithm Modules*. Defined below are the finite state machine and module for the manual error intervention, the diagram for which is shown in Figure 9:

$$F_M = (X_{F_M}, T_{F_M}, R_{F_M}, \tau_{F_{CM}}, x_{0_{F_M}}, x_{dp_{F_M}}, X_{M_{F_M}})$$

where

X_{F_M} = {idle, manual_control}
T_{F_M}= {1.error, 2.auto}
R_{F_M} = {1.error_done, 2.manual}
τ_{F_M} = {(1.error, [ ], idle, manual_control, [2.manual] , [ ]),
(2.auto, [ ], manual_control, idle, [1.error_done] , [ ])}
$x_{0_{F_M}}$ = idle

$x_{dp_{F_M}}$ = dump
$X_{M_{F_M}}$ = idle

Defined below is the module for the manual error intervention:

$$M = (F_M, \{S_{M_1}\})$$

where

F_M = Described above.
S_{M_1} = {[1.error, 2.auto], [1.error_done, 1.manual]}

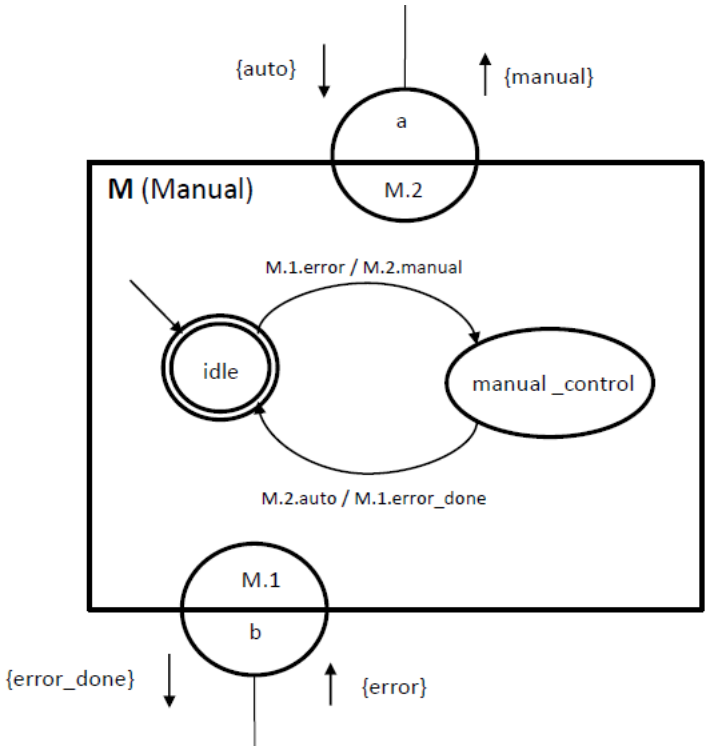

Figure 9. Manual error intervention module for snubbing operations.

6.6.4 Sub Level

The Sub Level of control contains the control modules which are specific to each piece of equipment. These modules take commands from the Low Level modules and

send the appropriate command to its corresponding equipment. Defined below are the finite state machine and module for the catcher's mitt, the diagram for which is shown in Figure 10:

$$F_{CM} = (X_{F_{CM}}, T_{F_{CM}}, R_{F_{CM}}, \tau_{F_{CM}}, x_{0_{F_{CM}}}, x_{dp_{F_{CM}}}, X_{M_{F_{CM}}})$$

where

$X_{F_{CM}}$ = {retracted, extended}
$T_{F_{CM}}$= {1.extend, 1.retract}
$R_{F_{CM}}$ = {1.extend_done, 1.retract_done}
$\tau_{F_{CM}}$ = {(1.extend, [ ], retracted, extended, [1.extend_done] , [ ]),
(1.retract, [ ], retracted, retracted, [1.retract_done] , [ ]),
(1.extend, [ ], extended, extended, [1.extend_done] , [ ]),
(1.retract, [ ], extended, retracted, [1.retract_done] , [ ]),
(1.retract, [ ], retracted, retracted, [1.retract_done] , [ ])}
$x_{0_{F_{CM}}}$ = retracted
$x_{dp_{F_{CM}}}$ = dump
$X_{M_{F_{CM}}}$ = retracted

Defined below is the equipment module for the catcher's mitt:

$$CM = (F_{CM}, \{S_{CM_1}\})$$

where

F_{CM} = Described above.
S_{CM_1} = {[1.extend, 1.retract], [1.extend_done, 1.retract_done]}

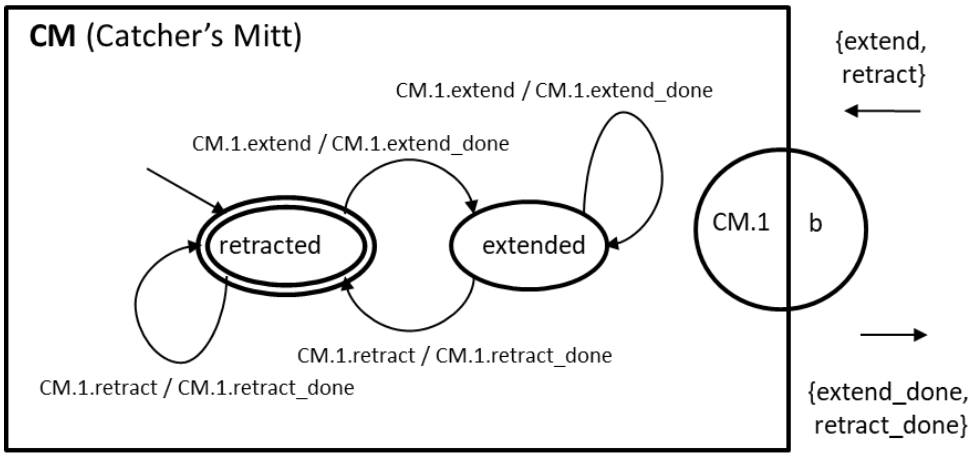

Figure 10. Catcher's mitt module in snubbing operations.

6.7 COLLISION DETECTION

Both the heuristics and bounding box methods to collision detection are good starting approaches to collision detection in snubbing operations. For ABB, the global reference frame for the bounding boxes will be located at the well center at the rig floor, as it is a fixed point that is integral to snubbing operations. Each piece of equipment will have a local reference frame attached to the center of the piece of equipment, and the equipment's bounding box will be defined with respect to the local reference frame. As with the approach in the case study, all equipment movement will be calculated with respect to the global reference frame, for detection of overlapping bounding boxes prior to the occurrence of a collision.

6.8 VERIFICATION AND VALIDATION

After the design of the supervisory controller is complete, it is necessary to verify the correctness of the controlled system model, and test the behavior of the system via model checking and simulations.

Model checking of the snubbing supervisory controller should be based on a finalized list of the LTL specifications described in Appendix F – Snubbing Automation Formal Specifications to verify the controlled system operates safely and behaves as desired.

The first phase of simulation for evaluation of the snubbing controller performance will be for error-free snubbing operations with original snubbing equipment. The second phase of simulation will be to evaluate the ability of the controller to detect common snubbing errors. The primary error cases which will be explored are as follows:

- End of pipe not caught by catcher's mitt during operations to transport pipe to and from the well center.
- Failure of tongs dies.
- Annular BOP leak.
- Overly tight joints while breaking connections.
- Stabbing plate misalignment.

If modifications to the controller design need to be made as a result of simulation observations, the error-free simulation will be rerun to ensure error-free operations have not been affected by the modifications. The final phase of simulation is to explore the case of introducing a new piece of equipment, for instance a gantry robot, to the snubbing unit. The capabilities of the gantry robot will be assessed to understand if and where modifications to the current structure must be made before simulation of the error-free and common error cases are done. Success of the controller feasibility will be evaluated through its ability to perform the desired operations for all simulations.

6.9 SUMMARY

The proven concept from the case study reflects the feasibility of modMFSM application to drilling and snubbing operations, as shown in Figure 6, which presents the modules for snubbing operations and how it compares to those from the conveyor belt drilling system case study. The system components have been identified, and a process narrative has been defined for snubbing operations. Preliminary LTL system specifications have been defined and initial designs of select snubbing operational modes are to be finalized in future research.

Chapter 7: Conclusions and Future Work

The completed work focused on definition of the extended modular finite state machine controller (modMFSM) structure, implementation of the modMFSM structure to a simplified conveyor belt drilling system, definitions of system equipment and environment for snubbing operations, and preliminary designs for the vending machine toy problem supervisory controller, found in Appendix G – Snubbing Modular Controller Preliminary Designs.

7.1 Conclusions

The contributions of this research to drilling automation supervisory control research is in providing a feasible framework for supervisory controllers for drilling operations and its auxiliary operations, while maintaining safe operations, overcoming the challenges of interoperability, reacting to environment cues, and allowing for easy modification and functional scalability.

The modMFSM structure allows for the augmentation of the modular finite state machine structure to adapt to complex systems and their uncertain environments. The first extension of the modular finite state machine structure is to explore the use of various models inside modules, specifically the inclusion of guards in the trigger / response finite state machines for response to changing environment and system variables.

The modMFSM structure also explores the feasibility of including modules which encompass physics-based models rather than discrete event models processes or hardware. In the snubbing case study, collision detection is implemented. Proving the feasibility of this implementation provides a stepping stone for inclusion of other physics-

based models environment in supervisory control of complex systems, for instance, allowing for kick detection in drilling or detection of alien objects in manufacturing.

Additionally, the modMFSM framework allows for automatic reconfiguration of the controller and automatic synthesis of filters and modules to replace missing equipment. The plug-and-play quality of the controller architecture also allows for easy incorporation of new equipment with minimal modification of the original controller.

7.2 FUTURE WORK

Future work of this research is to verify the formal correctness of the controlled simplified system model, and test for compositional correctness. Additionally, robustness of the system to input order can also be tested using the formal verification procedure presented and applied to ECA MFSMs by Allen et al. (Allen et al., 2012). After verification of the simplified system, the next step would be to finalize, validate, and verify the snubbing supervisory controller.

Appendices

APPENDIX A – SNUBBING UNIT EQUIPMENT

Snubbing units exist as both standalone and rig-assist units. This section discusses the general machinery used by all snubbing units.

Figure 11. Superior mechanized, standalone snubbing unit.

Similar to drilling operations, snubbing operations require a hoisting device, called an elevator, to run pipe into and out of the well during pipe-heavy operations. In snubbing, this process is called stripping. The elevator also raises and lowers pipe when pipe connections are made or broken. In addition to vertical movement, elevators include bails that extend and retract to allow for lateral movement of the pipe. While a standalone snubbing unit contains its own elevator, a rig-assist unit may utilize the elevator that exists as part of the hoisting system on the host drilling rig.

Figure 12. Floor of a workover unit where the yellow elevator is near the floor (Shengji Group, 2016).

Pipe is transported between its storage location in vertical or horizontal racks and the floor of the snubbing unit using pipe handling equipment. A variety of mechanized pipe handling equipment can be used, including catwalks for horizontal to vertical pipe handling, and racking systems like NOV's Hydraracker systems for transportation from vertical pipe racks. NOV's Hydraracker systems are able securely position new pipe ready for a connection to be made with the pipe in the well. However, if the pipe handling system requires the elevator to assist in bringing the pipe to the well center, a catcher's mitt in the snubbing unit is used to catch the new pipe as it is hoisted away from the pipe handling system and brought toward the well center.

Figure 13. a) Catwalk in action on a Superior snubbing unit (left); b) NOV Hydraracker (Hsieh & Vigh, 2017) (right).

Figure 14. Catcher's mitt extended and ready to catch the new pipe coming from the catwalk.

In order to connect or disconnect pipe as it is run into and out of the well, equipment such as hydraulically powered tongs apply the required torque to make or break the pipe-to-pipe connection.

Figure 15. NOV Hydraulic Power Tongs (National Oilwell Varco, 2018a)

In regular drilling operations, a device called slips grip and hold the drill string to the rig floor, preventing the drill string from falling into the well. For snubbing operations, slips serve two purposes: 1) hold the drill string stationary, 2) assist in pushing/pulling the drill string in/out of the well. The slips that hold the drill string stationary are called stationary slips and are located on the floor of the unit. Two sets of stationary slips exist, one set for pipe-light operations, and the other set for pipe-heavy operations. The slips that assist in snubbing the drill string are called the travelling slips. The travelling slips are located on the hydraulically powered snubbing jack, which moves the travelling slips vertically, applying the necessary snubbing force to run pipe in and out of the well. Similar to the stationary slips, the travelling slips also exist in two sets for pipe-light and pipe-heavy operations.

Figure 16. Snubbing jack near maximum height as it snubs out pipe.

Figure 17. Slips for snubbing operations (Westco International Consulting, Inc, 2017).

Since snubbing operations occur while the well is pressurized, well control equipment is crucial. The primary components of the basic snubbing well control equipment stack are: an annular blowout preventer (BOP), two snubbing pipe ram blowout preventers (BOPs), safety pipe ram BOPs, safety blind ram BOPs, an bleed-off line, and a pump-in line. BOPs prevent well fluids from uncontrolled travel to surface and allow the crew to maintain control on the wellbore pressure. The annular BOP, which sits at the top of the snubbing well control equipment stack, contains a rubber element that seals around the drill string and adjusts its seal as pipe joints, and tools pass through it during stripping and snubbing operations. The annular BOP remains closed throughout the snubbing operations. The snubbing pipe ram BOPs are manufactured to fit either one pipe diameter or a small range of pipe diameters (Mitchell & Miska, 2011). Therefore, the pipe ram BOPs seal around the drill pipe only, and can handle higher pressures than the annular BOP. They generally remain open if the wellbore pressure is below the working pressure of the annular BOP. However, if the wellbore pressure exceeds the working pressure of the annular BOP, at least one pipe ram BOP must be closed around the pipe at all times to ensure a secure pressure barrier as pipe is run into and out of the well. However, since the pipe ram BOPs can only fit a small range of diameters at most, special procedures are necessary to pass pipe joints and tools through the pipe ram BOPs. For snubbing in procedures, the upper pipe ram starts off closed with the lower pipe ram open. When the joint or tool reaches the upper pipe ram, the lower pipe ram is closed, and the bleed-off line opened to bleed off the pressure between the rams. The upper pipe ram is then opened, and the pipe lowered until the joint or tool is at the lower pipe ram. The upper pipe ram is then closed, and the pump-in line is opened to pressurize the section between the rams. Once the pressure in that section equals that of the well, the lower pipe ram is opened, and snubbing operations continue. The reverse procedures are

implemented for snubbing out operations. The safety pipe rams and the safety blind rams generally remain open during operations and are only closed in case the snubbing rams malfunction. Although the frac valve is part of the wellhead and not part of the snubbing stack, it is an integral piece of well control equipment. The frac valve sits below the snubbing well control equipment stack, and at the top of the wellhead. It remains open during snubbing operations, and is closed when it is necessary to shut in the well.

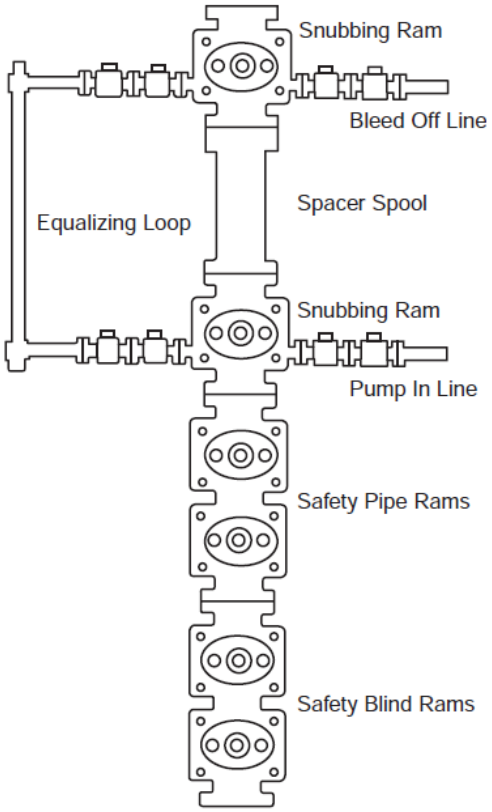

Figure 18. Basic snubbing well control equipment stack (Grace, 2003). Annular BOP not shown.

APPENDIX B – CASE STUDY CONTROLLER DESIGN

The design for the modular supervisory controller for the conveyor belt drilling system case study can be found in the supplemental file labeled "Appendix_B_Case_Study_Controller_Design.pdf."

APPENDIX C – CASE STUDY SIMULATION TRANSITIONS AND CODE OUTPUT

The expected transitions of each simulation case for the conveyor belt drilling system case study can be found in the supplemental file labeled "Appendix_C_Case_Study_Expected_Transitions.xlsx."

The simulation code output for each case can be found in the following supplemental files: "Appendix_C_Case_1_Simulation_Code_Output.pdf," "Appendix_C_Case_2_Simulation_Code_Output.pdf," and "Appendix_C_Case_3_Simulation_Code_Output."

APPENDIX D – PROCESS NARRATIVE FLOWCHARTS FOR SNUBBING

The mechanized snubbing operations are captured in a series of process narrative flowcharts. These flowcharts can be found in the supplemental file labeled "Appendix_D_Process_Narrative_Flowcharts_for_Snubbing.pptx."

APPENDIX E – SNUBBING AUTOMATION SENSORS

Table 3 below presents the system variable necessary for automating snubbing operations and the status of its corresponding sensor. From the table, it can be seen that at least eighteen sensors are needed for autonomously bringing new pipe to the well center. From left to right, the columns describe the variable observed, whether sensor hardware exists and its type, suggested sensor hardware if not already available, and the current sensing method for the variable.

Table 3. Sensors necessary for automated snubbing operations.

Variable	**Sensor Hardware**	**Suggested Hardware**	**Current Method**
Stabbing Plate Alignment	None	Camera	Visual / Manual
Well Pressure	Casing Pressure Gauge	N/A	Sensor
Joint Counter	None	N/A (Build into controller)	Visual / Manual
Stripping Bit Depth	Block Position	N/A	Sensor
Height of Pipe for Connection			Sensor
Elevator Position			Visual / Manual
Tripping Speed			Sensor
TD/Plug Reached			Visual / Manual
Pull Test for Slips Bite Before Balance Point	None	Block Position or Snubbing Jack Position	Visual / Manual
Snubbing Bit Depth	Snubbing Jack Position (Existence Uncertain)	Snubbing Jack Position (Existence Uncertain)	Sensor
Height of Pipe for Connection			Visual / Manual
Snubbing Jack Position			Visual / Manual
Snubbing Jack Speed			Visual / Manual
Plug Reached			Visual / Manual
Pipe Weight			Visual / Manual
Elevator Closed on Pipe	Built-In Open/Close Indicator	N/A	Visual / Manual

(Table 3 Continued)

Variable	**Sensor Hardware**	**Suggested Hardware**	**Current Method**
Catwalk Height	Built-In Sensor (Possible Limit Switch)	N/A	Sensor
Elevator Height at Pipe	None	Camera	Visual / Manual
Catcher's Mitt (Extended/Retracted)	Built-In Indicator (Existence Uncertain)	N/A	Visual / Manual
Tongs (Extended/Retracted)	Built-In Indicator - Well center is set point	N/A	Sensor
Pipe in Catcher's Mitt	None	Contact or Proximity Sensor	Visual / Manual
Snub Force (Applied)	Snub Gauge	N/A	Sensor
Pressure between RAMs	Pressure Gauge	N/A	Sensor
Tongs Torque	Built-In Torque Sensor	N/A	Sensor
Pipe End in Catwalk	None	Catwalk Camera	Visual / Manual
Pipe in Skate Basket			Visual / Manual
Tongs Dies Failure	Built-In Torque Sensor; Motor rotation encoder on pipe (for connections)	N/A	Sensor
Connection Made			Sensor
BHA Reached	None	Catwalk Camera (BHA nipple or flagging pup)	Visual / Manual
Hookload for Pull Test (~14klbs)	Hookload	N/A	Sensor / Visual
Slips at Tool or Connection	None	Camera	Visual / Manual
Connection Position Relative to RAMS	None	N/A (Calculations)	Manual
Buoyant Force	None	N/A	
Friction Force	None	N/A	

APPENDIX F – SNUBBING AUTOMATION FORMAL SPECIFICATIONS

The following table presents the formal specifications for snubbing operations derived from the process narrative. Modifications are expected to be needed to include requirements for module-to-module behavior specifications.

Table 4. Formal specifications for system requirements in operations for bringing new pipe to the well center

Type	**Requirement Behavior**	**Linear Temporal Logic**
Liveness	Number of pipes in well will eventually be desired number of pipes	**GF** (Num_Pipes_In = Des_Pipes)
Liveness	Number of pipes out well will eventually be desired number of pipes	**GF** (Num_Pipes_Out = Des_Pipes)
Invariant	Use annular BOP if P_well < working pressure of annular BOP	((P_well < P_BOP_ann_working) **->** (Use annular BOP)) ^ ((P_well >= P_BOP_ann_working) **->** (Use RAMs))
Safety	F_snub > F_well + F_buoyant + F_friction - W_pipe	**G** (F_snub > F_well + F_buoyant + F_friction - W_pipe)
Safety	F_snub < Buckling Force of Pipe	**G** (F_snub < F_Buckling_Crit/SF)
Safety	F_snub < Collapse Force of Pipe	**G** (F_snub < F_Collapse_Crit/SF)
Safety	F_snub < Burst Force of Pipe	**G** (F_snub < F_Burst_Crit/SF)
Safety	Snubbing Speed < Snub Speed Limit (Surge/Swab)	**G** (Speed_snub < Speed_snub_max)
Safety	Tripping Speed < Trip Speed Limit (Surge/Swab)	**G** (Speed_trip < Speed_trip_max)
Invariant	Joints in after Balance Point < 20 Implies Elevator Following	(Jts_In_After_Bal_Pt < 20) **->** (Elevator_Following_Trip)

(Table 4 Continued)

Invariant	Joints out after Balance Point < 10 Implies Elevator Following	(Jts_Out_After_Bal_Pt < 10) -> (Elevator_Following_Snub)
Invariant	In Direction AND Feet Pipe in Well < Feet In at Balance Point - Tolerance Implies SIPL	(Dir_In ^ (Pipe_feet_in < Pipe_at_Bal_Pt + Tol)) -> (SIPL)
Invariant	In Direction AND Feet Pipe in Well > Feet In at Balance Point + Tolerance Implies SIPH	(Dir_In ^ (Pipe_feet_in > Pipe_at_Bal_Pt + Tol)) -> (SIPH)
Invariant	Out Direction AND Feet Pipe in Well < Feet In at Balance Point - Tolerance Implies SOPL	(Dir_Out ^ (Pipe_feet_in < Pipe_at_Bal_Pt + Tol)) -> (SOPL)
Invariant	Out Direction AND Feet Pipe in Well > Feet In at Balance Point + Tolerance Implies SOPH	(Dir_Out ^ (Pipe_feet_in > Pipe_at_Bal_Pt + Tol)) -> (SOPH)
Safety	Snubbing Jack Height <= Maximum Safe Jack Height	**G** (Jack_Height < Max_Jack_Safe_Height)
Safety	Snubbing Jack Height >= Minimum Safe Jack Height	**G** (Jack_Height > Min_Jack_Safe_Height)
None	Feet Pipe Moved > Frequency for Filling Hole with Hydraulic Fluid Implies Fill Hole with Hydraulic Fluid	(Pipe_Moved > Freq_Fill_with_Hydraulic_Fluid) -> **X** (Fill Hole with Hydraulic Fluid)
Safety	Do not close empty RAMs	**G** (RAM_empty -> RAM_open)
Safety	Do not close slips on a tool or connection	**G** (((Jack_Height < Tool_Joint_Max) ^ (Jack_Height > Tool_Joint_Min)) -> Slips_Open)

(Table 4 Continued)

Safety	If catcher's mitt or tongs extended, elevator position must be above safe lower limit, and snubbing jack must be at the minimum safe jack height.	**G** ((Mitt_Extended **v** Tongs_Extended) **->** ((Elevator_Height >= Safe_Lower_Limit) ^ (Jack_Height = Min_Jack_Safe_Height)))
Safety	If elevator position below safe lower limit or snubbing jack above safe lower limit, catcher's mitt and tongs must be retracted.	**G** (((Elevator_Height < Safe_Lower_Limit) **v** (Jack_Height >Min_Jack_Safe_Height)) **->** (**!**Mitt_Extended ^ **!**Tongs_Extended))
Safety	If snubbing jack not at safe lower limit, elevator position must be above safe lower limit, and catwalk cannot push up pipe.	**G** ((Jack_Height > Min_Jack_Safe_Height) **->** ((Elevator_Height > Safe_Lower_Limit) ^ (Skate_Basket < Max_Skate_Height)))
Safety	Elevator height should always remain more than a margin above the snubbing jack height.	**G** (Elevator_Height > Jack_Height + Margin)
Safety	If connection made or connection not broken, catcher's mitt must not be extended.	**G** ((Connection_Made **v** **!**Connection_Broken) **->** **!**Mitt_Extended)

APPENDIX G – SNUBBING MODULAR CONTROLLER PRELIMINARY DESIGNS

The preliminary designs of the modular supervisory controller for the following operation modes are complete:

- Snubbing New Pipe (SNP) – Moves pipe from the pipe rack to the well center.
- Snubbing Rack Pipe (SRP) – Moves pipe from the well center to the pipe rack.
- Snub In Pipe Heavy (SIPH) – Snubs pipe into the well center under pipe-heavy conditions.
- Trip In Pipe Heavy (TIPH) – Trips pipe into the well center under pipe-heavy conditions.

These preliminary designs are expected to be modified before error-free simulation to include collision detection modules. Additionally, modifications may be made as necessary to accommodate additional assumptions for system simplification. The completed preliminary designs can be found in the supplemental file labeled "Appendix_G_Modular_Controller_Preliminary_Designs.pdf."

References

Allen, L. V., Goh, K. M., & Tilbury, D. M. (2012). Input Order Robustness: Definition, Verification Procedure, and Examples. *IEEE Transactions on Automation Science and Engineering*, *9*(1), 3–15. https://doi.org/10.1109/TASE.2011.2163509

Almeida, E. E., Luntz, J. E., & Tilbury, D. M. (2007). Event-condition-action systems for reconfigurable logic control. *IEEE Transactions on Automation Science and Engineering*, *4*(2), 167–181. https://doi.org/10.1109/TASE.2006.880857

Almeida, E. T., Luntz, J. E., & Tilbury, D. M. (2005). Modular finite state machines implemented as event-condition-action systems. *IFAC Proceedings Volumes*, *38*(1), 373–378. https://doi.org/10.3182/20050703-6-CZ-1902.00346

Asarin, E., Maler, O., & Pnueli, A. (1994). Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems II* (pp. 1–20). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-60472-3_1

Ayling, L. J., Jenner, J. W., & Neffgen, J. M. (2003). Seabed Located Drilling Rig - ITF Pioneer Project. Presented at the Offshore Technology Conference, Offshore Technology Conference. https://doi.org/10.4043/15328-MS

Basile, F., Chiacchio, P., & Coppola, J. (2012). A Hybrid Model of Complex Automated Warehouse Systems—Part I: Modeling and Simulation. *IEEE Transactions on Automation Science and Engineering*, *9*(4), 640–653. https://doi.org/10.1109/TASE.2012.2215322

Beyer, D., & Lemberger, T. (2017). Software Verification: Testing vs. Model Checking. In O. Strichman & R. Tzoref-Brill (Eds.), *Hardware and Software: Verification and Testing* (pp. 99–114). Springer International Publishing.

Brugman, J. D. (1987). Automated Pipe Handling: A Fresh Approach. Presented at the SPE/IADC Drilling Conference, Society of Petroleum Engineers. https://doi.org/10.2118/16065-MS

Calderoni, A., & Cercato, M. (2015). Drillmec Ahead: A New Generation of Fully Automated Drilling Rig Forward Designed to Meet Highest HSE Standards and Drilling Efficiency. Presented at the Offshore Mediterranean Conference and Exhibition, Offshore Mediterranean Conference. Retrieved from https://www.onepetro.org/conference-paper/OMC-2015-459?sort=&start=0&q=drillmec&from_year=&peer_reviewed=&published_between=&fromSearchResults=true&to_year=&rows=10#

Cao, D., & Moralez, N. (2016). Spare Parts Strategy Based on RAM Analysis to Extend the Life of Drilling Supervisory and Control Systems: A Case Study. Presented at the IADC/SPE Drilling Conference and Exhibition, Society of Petroleum Engineers. https://doi.org/10.2118/178759-MS

Cassandra, A. R. (1998). A Survey of POMDP Applications. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–97)* (pp. 472– 480).

Cassandras, C. G., & Lafortune, S. (2011). *Introduction to discrete event systems*. New York; London: Springer.

Chang, J.-W., Choi, Y.-K., Kim, M.-S., & Wang, W. (2011). Computation of the minimum distance between two Bézier curves/surfaces. *Computers & Graphics*, *35*(3), 677–684. https://doi.org/10.1016/j.cag.2011.03.025

Chen, S. Y. (2012). Kalman Filter for Robot Vision: A Survey. *IEEE Transactions on Industrial Electronics*, *59*(11), 4409–4420. https://doi.org/10.1109/TIE.2011.2162714

Chen, Y. L., & Lin, F. (2000). Modeling of discrete event systems using finite state machines with parameters. In *Proceedings of the 2000. IEEE International Conference on Control Applications. Conference Proceedings (Cat. No.00CH37162)* (pp. 941–946). https://doi.org/10.1109/CCA.2000.897591

De, A., Silin, D. B., & Patzek, T. W. (2000). Waterflood Surveillance and Supervisory Control. Presented at the SPE/DOE Improved Oil Recovery Symposium, Society of Petroleum Engineers. https://doi.org/10.2118/59295-MS

Dunham, C. L. (1987). Supervisory Control of Beam Pumping Wells. Presented at the SPE Production Operations Symposium, Society of Petroleum Engineers. https://doi.org/10.2118/16216-MS

Endsley, E. W. (2004). *Modular finite state machines for logic control: Theory, verification and applications to reconfigurable manufacturing systems.* University of Michigan. Retrieved from https://deepblue.lib.umich.edu/handle/2027.42/124052

Fowze, F., & Yavuz, T. (2016). Specification, verification, and synthesis using extended state machines with callbacks. In *2016 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)* (pp. 95–104). https://doi.org/10.1109/MEMCOD.2016.7797752

Gaudin, B., & Deussen, P. H. (2007). Supervisory Control on Concurrent Discrete Event Systems with Variables. In *2007 American Control Conference* (pp. 4274–4279). https://doi.org/10.1109/ACC.2007.4282808

Gaushell, D. J., & Darlington, H. T. (1987). Supervisory control and data acquisition. *Proceedings of the IEEE*, *75*(12), 1645–1658. https://doi.org/10.1109/PROC.1987.13932

Genrich, H. J. (1986). Predicate/Transition Nets. In *Petri Nets: Central Models and Their Properties* (pp. 207–247). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-47919-2_9

Grace, R. D. (2003). *Blowout and Well Control Handbook - 1st Edition* (1st ed.). Gulf Professional Publishing. Retrieved from https://ebookcentral-proquest-com.ezproxy.lib.utexas.edu/lib/utxa/detail.action?docID=294007

Han, R., Ashok, P., Pryor, M., van Oort, E., Scott, P., Reese, I., & Hampton, K. (2017). Real-Time Borehole Condition Monitoring using Novel 3D Cuttings Sensing Technology. Presented at the SPE/IADC Drilling Conference and Exhibition, Society of Petroleum Engineers. https://doi.org/10.2118/184718-MS

Hansen, J., & Witte, L. E. (1999). *US5988299 A*. Retrieved from http://www.google.com/patents/US5988299

Hegde, C., Awan, O., & Wiemers, T. (2018). Application of Real-time Video Streaming and Analytics to Breakdown Rig Connection Process. Presented at the Offshore Technology Conference, Offshore Technology Conference. https://doi.org/10.4043/28742-MS

Hill, R. C., Cury, J. E. R., de Queiroz, M. H., Tilbury, D. M., & Lafortune, S. (2010). Multi-level hierarchical interface-based supervisory control. *Automatica*, *46*(7), 1152–1164. https://doi.org/10.1016/j.automatica.2010.04.002

Hill, Richard Charles. (2008). *Modular verification and supervisory controller design for discrete -event systems using abstraction and incremental construction* (Ph.D.). University of Michigan, United States -- Michigan. Retrieved from https://search.proquest.com/docview/304574278/abstract/4BD5A51ADDFB4C8EPQ/1

Huisman. (n.d.). *LOC 400 Drilling Rig: Huisman Product Brochure*. Retrieved from https://www.huismanequipment.com/documenten/brochure_loc_400_26102015_mail.pdf

Ito, T. (1997). Emerging Technology-A Portable SCADA System. Presented at the Annual Technical Meeting, Petroleum Society of Canada. https://doi.org/10.2118/97-48

Jacobs, T. (2015). Automated Drilling Technologies Showing Promise. *Journal of Petroleum Technology*, *67*(06), 50–55. https://doi.org/10.2118/0615-0050-JPT

Jiménez, P., Thomas, F., & Torras, C. (2001). 3D collision detection: a survey. *Computers & Graphics*, *25*(2), 269–285. https://doi.org/10.1016/S0097-8493(00)00130-8

Khorasgani, H., & Biswas, G. (2018). Structural Fault Detection and Isolation in Hybrid Systems. *IEEE Transactions on Automation Science and Engineering*, *15*(4), 1585–1599. https://doi.org/10.1109/TASE.2017.2749447

Kockara, S., Halic, T., Iqbal, K., Bayrak, C., & Rowe, R. (2007). Collision detection: A survey. In *2007 IEEE International Conference on Systems, Man and Cybernetics* (pp. 4046–4051). https://doi.org/10.1109/ICSMC.2007.4414258

Kurz, K. (2013). *Possible Implementation of Automated Systems in Drilling Rig Design*. Montanuniversität Leoben.

Leduc, R. J., Dai, P., & Song, R. (2009). Synthesis Method for Hierarchical Interface-Based Supervisory Control. *IEEE Transactions on Automatic Control*, *54*(7), 1548–1560. https://doi.org/10.1109/TAC.2009.2022101

Lee, C. W., & Won, K. S. W. (2013). Running Casing in Stands with Automate Casing Drive System - Case Studies. Presented at the International Petroleum Technology Conference, International Petroleum Technology Conference. https://doi.org/10.2523/IPTC-16713-MS

Lee, J. S., Zhou, M., & Hsu, P. L. (2007). A Petri-Net Approach to Modular Supervision With Conflict Resolution for Semiconductor Manufacturing Systems. *IEEE Transactions on Automation Science and Engineering*, *4*(4), 584–588. https://doi.org/10.1109/TASE.2007.905995

Lennartson, B., Basile, F., Miremadi, S., Fei, Z., Hosseini, M. N., Fabian, M., & Åkesson, K. (2014). Supervisory Control for State-Vector Transition Models—A Unified Approach. *IEEE Transactions on Automation Science and Engineering*, *11*(1), 33–47. https://doi.org/10.1109/TASE.2013.2291115

Lennartson, B., Bengtsson, K., Wigström, O., & Riazi, S. (2016). Modeling and Optimization of Hybrid Systems for the Tweeting Factory. *IEEE Transactions on Automation Science and Engineering*, *13*(1), 191–205. https://doi.org/10.1109/TASE.2015.2480010

Lipka, R., Paška, M., & Potužák, T. (2014). Simulation testing and model checking: a case study comparing these approaches. In I. Majzik & M. Vieira (Eds.), *Software Engineering for Resilient Systems: 6th International Workshop, SERENE 2014 Budapest, Hungary, October 15-16, 2014 Proceedings* (Vol. 8785). Springer International Publishing. Retrieved from https://link-springer-com.ezproxy.lib.utexas.edu/chapter/10.1007/978-3-319-12241-0_9

Loeyning, T. (2017). Automated Catwalk Machine for Land Rigs. Presented at the Abu Dhabi International Petroleum Exhibition & Conference, Society of Petroleum Engineers. https://doi.org/10.2118/188752-MS

MacGregor, C. (n.d.). The Rig of the Future: Making Every Well Your Best Well. Retrieved May 4, 2018, from https://www.software.slb.com

Macpherson, J. D., Wardt, D., P, J., Florence, F., Chapman, C., Zamora, M., … Iversen, F. (2013). Drilling-Systems Automation: Current State, Initiatives, and Potential Impact. *SPE Drilling & Completion*, *28*(04), 296–308. https://doi.org/10.2118/166263-PA

Malik, R., & Teixeira, M. (2016). Modular supervisor synthesis for extended finite-state machines subject to controllability. In *2016 13th International Workshop on*

Discrete Event Systems (WODES) (pp. 91–96). https://doi.org/10.1109/WODES.2016.7497831

McLeod, R. W. (2015). Automation and supervisory control. In *Designing for Human Reliability* (pp. 159–169). Boston: Gulf Professional Publishing. https://doi.org/10.1016/B978-0-12-802421-8.00009-6

Nabors. (2018). Automated Drilling Floor Systems - Automated Tubular and Tool Handling Equipment. Retrieved May 4, 2018, from https://www.nabors.com/equipment/automated-floor-systems

Nabors Industries Ltd. (2018). Nabors SmartRig™ drilling systems. Retrieved November 7, 2018, from https://www.nabors.com/rigs/onshore-rigs/nabors-smartrig

National Oilwell Varco. (2018a). NOVOS: Reflexive Drilling System. Retrieved May 4, 2018, from http://www.nov.com/NOVOS.aspx

National Oilwell Varco. (2018b). NOVOS: Step into the rig of the future. Retrieved February 16, 2018, from https://www.nov.com/Segments/Rig_Systems/Land/Control_and_Advisory_Systems/Integrated_Drilling_Control_Systems/NOVOS.aspx

Ngo, Q. H., & Seow, K. T. (2014). Command and Control of Discrete-Event Systems: Towards Online Hierarchical Control Based on Feasible System Decomposition. *IEEE Transactions on Automation Science and Engineering*, *11*(4), 1218–1228. https://doi.org/10.1109/TASE.2013.2278975

Nishi, T., Watanabe, Y., & Sakai, M. (2018). An Efficient Deadlock Prevention Policy for Noncyclic Scheduling of Multicluster Tools. *IEEE Transactions on Automation Science and Engineering*, *15*(4), 1677–1691. https://doi.org/10.1109/TASE.2017.2771751

Niven, R. D. (1971). A new era in supervisory control. *Journal of Canadian Petroleum Technology*, *10*(04). https://doi.org/10.2118/71-04-02

Nuzzo, P., Finn, J. B., Iannopollo, A., & Sangiovanni-Vincentelli, A. L. (2014). Contract-based design of control protocols for safety-critical cyber-physical systems. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)* (pp. 1–4). https://doi.org/10.7873/DATE.2014.072

Ornas, J. I. (2010). Closed-Loop Control for Decision-Making Applications in Remote Operations. Presented at the IADC/SPE Drilling Conference and Exhibition, Society of Petroleum Engineers. https://doi.org/10.2118/126907-MS

Prati, T. J., Farines, J. M., & de Queiroz, M. H. (2015). Automatic test of safety specifications for PLC programs in the Oil and Gas Industry. *IFAC-PapersOnLine*, *48*(6), 27–32. https://doi.org/10.1016/j.ifacol.2015.08.005

Prebeau-Menezes, L. J. (2013). *Snubbing Field Operations-Potential Trapped Air and Explosive Hydrocarbon Mixtures on Surface* (PhD Thesis). University of Calgary.

Ramadge, P. J. G., & Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of the IEEE, 77*(1), 81–98. https://doi.org/10.1109/5.21072

Saadallah, N., Meling, H., & Daireaux, B. (2011). Modeling a drilling control system, as a Discrete-Event-System. In *2011 International Conference on Communications, Computing and Control Applications (CCCA)* (pp. 1–5). https://doi.org/10.1109/CCCA.2011.6031461

Saadallah, Nejm. (2013). *Scheduling drilling processes with Petri nets*. University of Stavanger, Faculty of Science and Technology, Stavanger.

Saez, M., Maturana, F. P., Barton, K., & Tilbury, D. M. (2018). Real-Time Manufacturing Machine and System Performance Monitoring Using Internet of Things. *IEEE Transactions on Automation Science and Engineering*, *15*(4), 1735–1748. https://doi.org/10.1109/TASE.2017.2784826

Schlumberger Limited. (2018). OnTrack Integrated Drilling Controls System. Retrieved November 8, 2018, from https://www.products.slb.com/rig-equipment/cabins-and-controls/ontrack-integrated-drilling-controls-system

Söderberg, A., & Johansson, R. (2013). Safety contract based design of software components. In *2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 365–370). https://doi.org/10.1109/ISSREW.2013.6688922

Superior Energy. (2016). Superior Energy - Automated Completion Rigs. Retrieved February 15, 2018, from https://superiorenergy.com/brands/SPN-well-services/CATS-automated-completion-rigs/

Szpyrka, M., Biernacka, A., & Biernacki, J. (2014). Methods of Translation of Petri Nets to NuSMV Language. In *Proc. Int. Workshop Concurrency Specification Program* (pp. 245–256). Retrieved from /paper/Methods-of-Translation-of-Petri-Nets-to-NuSMV-Szpyrka-Biernacka/9b0b630e101bd44e86b73771adeb3c9dcdfadcbb

Teixeira, M., Malik, R., Cury, J. E. R., & Queiroz, M. H. de. (2015). Supervisory Control of DES With Extended Finite-State Machines and Variable Abstraction. *IEEE Transactions on Automatic Control*, *60*(1), 118–129. https://doi.org/10.1109/TAC.2014.2337411

Thistle, J. G. (1996). Supervisory control of discrete event systems. *Mathematical and Computer Modelling*, *23*(11), 25–53. https://doi.org/10.1016/0895-7177(96)00063-5

Topcu, U. (2017). *LT Properties*. PowerPoint presentation.

Ugasciny, A., Chang, B. T., & Hampson, A. (2016). Cost Reduction on Onshore Rig Operation with Offline Stand Building System. Presented at the SPE Annual

Technical Conference and Exhibition, Society of Petroleum Engineers. https://doi.org/10.2118/181292-MS

West Group. (2015, June 25). An Historic Event. Retrieved January 12, 2018, from http://www.westgroup.no/home/-an-historic-event-

Wilson, N. D. (1971). East Rainbow Supervisory Control System. *Journal of Canadian Petroleum Technology*, *10*(03). https://doi.org/10.2118/71-03-05

Wu, N., Zhou, M., & Chu, F. (2008). A Petri Net-Based Heuristic Algorithm for Realizability of Target Refining Schedule for Oil Refinery. *IEEE Transactions on Automation Science and Engineering*, *5*(4), 661–676. https://doi.org/10.1109/TASE.2008.916737

Yang, Y., & Gohari, R. (2005). Embedded supervisory control of discrete-event systems. In *IEEE International Conference on Automation Science and Engineering, 2005.* (pp. 410–415). https://doi.org/10.1109/COASE.2005.1506804

Zheng, X., Julien, C., Kim, M., & Khurshid, S. (2017). Perceptions on the State of the Art in Verification and Validation in Cyber-Physical Systems. *IEEE Systems Journal*, *11*(4), 2614–2627. https://doi.org/10.1109/JSYST.2015.2496293

Zhu, M., & Brooks, R. R. (2009). Comparison of Petri Net and Finite State Machine Discrete Event Control of Distributed Surveillance Network. *International Journal of Distributed Sensor Networks*, *5*(5), 480–501. https://doi.org/10.1080/15501320903048753