Modeling and Performance Analysis of Distributed Systems with Collaboration Behaviour Diagrams

Toqeer Israr

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
for the Doctorate in Philosophy degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science (EECS)
Faculty of Engineering
University of Ottawa

ABSTRACT

The use of distributed systems, involving multiple components, has become a common industry practice. However, modeling the behaviour of such systems is a challenge, especially when the behavior consists of several collaborations of different parties, each involving possibly several starting (input) and ending (output) events of the involved components. Furthermore, the global behavior should be described as a composition of several sub-behaviours, in the following called collaborations, and each collaboration may be further decomposed into several sub-collaborations. We assume that the performance of the elementary sub-collaborations is known, and that the performance of the global behavior should be determined from the performance of the contained elementary collaborations and the form of the composition.

A collaboration, in this thesis, is characterized by a partial order of input and output events, and the performance of the collaboration is defined by the minimum delays required for a given output event with respect to an input event. This is a generalization of the semantics of UML Activities, where all input events are assumed to occur at the same time, and all output events occur at the same time. We give a semantic definition of the dynamic behavior of composed collaborations using the composition operators for control flow from UML Activity diagrams, in terms of partial order relationships among the involved input and output events. Based on these semantics, we provide formulas for calculating the performance of composed collaborations in terms of the performance of the sub-collaborations, where each delay is characterized by (a) a fixed value, (b) a range of values, and (c) a distribution (in the case of stochastic behaviours). We also propose approximations for the case of stochastic behavior with Normal distributions, and discuss the expected errors that may be introduced due to ignoring of

shared resources or possible dependencies in the case of stochastic behaviours. A tool has been developed for evaluating the performance of complex collaborations, and examples and case studies are discussed to illustrate the applicability of the performance analysis and the visual notation which we introduced for representing the partial-order relationships of the input and output events.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

Acknowledgments

Alhamdulillah! First and foremost, I would like to thank Allah (SWT) for allowing me to complete this work. "Nothing shall ever happen to us except what Allah has ordained for us." Quran 9:51

I would like to thank my supervisor Dr. Gregor v. Bochmann, for his invaluable support. Although I frequently resisted his suggestions (and usually I was wrong), his observations were almost always correct and this dissertation has benefited greatly due to his patience, motivation, enthusiasm, and immense knowledge. Thank you Dr. Bochmann!

I would like to acknowledge the financial, academic and technical support of University of Ottawa and its staff and allowing me the opportunity to pursue my research in such a great environment.

To my parents – the people I look up to and respect the most. Your unconditional love, support, and prayers gave me the strength to commence and complete this task. Thank you from the bottom of my heart! Junaid Bhai, Tauseef Bhai, Moeed Bhai & Shazima Baji, Arooj Baji, Huma – your love and support has kept me grounded to complete this work – Thank you for everything!

I would like to thank my children, Taha and Hadia, who have never known their father as anything but a student. Without their never ending love, it is unlikely I would have completed this task.

Finally, a special thanks to the person who never gave up on me – person who encouraged me when I started losing hope, made me laugh when I was sad and with whom I share fantastic memories with – Thank you Naima.

I also would like to thank Naveed Mamoon & family for their prayers, love and support.

Abbreviations

AD	Activity Diagram
API	Application Programming Interface
BPD	Business Process Diagram
BPMN	Business Process Model and Notation
CBD	Collaboration Behaviour Diagram
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
CTMC	Continuous Time Markov Chain
DAG	Directed Acyclic Graph
DEDS	Discrete Event Dynamic Systems
DES	Discrete Event Simulation
FD	Fixed Delays
FIFO	First In First Out
FSM	Finite State Machine
GQAM	Generic Quantitative Analysis Modeling
GSMP	Generalized Semi Markov Processes
GUI	Graphic User Interface
HTML	HyperText Markup Language
IBM	International Business Machines Corporation
IDE	Integrated Development Environment

IOA	Input Output Automata
J2EE	Java 2 Platform Enterprise Edition
LIFO	Last In First Out
LQN	Layered Queuing Network
MARTE	Modeling and Analysis of Real Time and Embedded Systems
MSC	Message Sequence Charts
NETD	Nominal Execution Time Delays
NoQ	Network of Queue
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
PAM	Performance Analysis Modeling
PAT	Performance Analysis Tool
PDF	Probability Density Function
PERT	Performance Evaluation and Review Technique
PN	Petri Nets
POIOA	Partial Order Input Output Automata
QM	Queuing Models
RD	Range of Delays
RFP	Request For Proposal
RSA	Rational Software Architect
RTES	Real–Time and Embedded Systems

SAM	Schedulability Analysis Modeling
SD	Stochastic Delays
SDL	Specification and Description Language
SDS	Simplified Data Structure
SMP	Semi Markov Process
SPN	Stochastic Petri Nets
SPT	UML Profile for Schedulability, Performance and Time
SSQ	Single Server Queue
TPN	Timed Petri Nets
UCM	Use Case Maps
UML	Unified Modeling Language
WS-BPEL	Web Services - Business Process Execution Language
WS-BPMN	Web Services - Business Process Model and Notation
WS-CDL	Web Services – Choreography Description Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XPDL	XML Process Definition Language

1. INTRODUCTION

1.1 Motivation

Representing and reasoning about the dynamic aspects of the world, primarily about actions and events and their associated performance, is a problem of interest to many different disciplines.

Many commercial systems, especially Web-based applications, are composed of a number of communicating components. These are often structured as distributed systems, with components running on different processors or in different processes. For example, a multi-tiered system might start with requests from Web Clients that flow through a "front-end" Web-Server and then to a Web "Application Server," which in turn makes calls to a "Database Server", and perhaps to additional servers (authentication, name service, credit-card authorization, customer relationship management, etc.).

When developing such distributed reactive systems, there is a need to model behavior both from a global system perspective and a local or component perspective. The global perspective specifies and analyzes the collaborative behavior of a distributed system in an abstract manner, while the local perspective identifies different system components along with their behaviour such that their interactions give rise to a behaviour satisfying the global perspective. Numerous methodologies have been defined to model such systems, such as UML Activity Diagrams [62], Use Case Maps (UCM) [13], the Process Definition Language (XPDL) [76], Business Process Model and Notation (BPMN) [64], Business Process Execution Language

(BPEL) [88], Web Services Choreography Description Language (WS-CDL) [83] and Petri Nets [7, 16, 17, 29].

Most, if not all, of these notations can potentially decompose a system into *sub-systems* and further into *sub-sub-systems*, where the final decomposition typically identifies a single component. However, quite often in many distributed systems, even the final decomposition involves multiple components. This is often referred as the *crosscutting* nature of distributed services [43, 72], i.e. services involve several collaborating components playing different roles, that each may participate in several services. By modeling according to the roles, their behaviour can be defined precisely, but the behaviour of a service is fragmented. To model the global behaviour of a service explicitly, an orthogonal view is needed where the collaborative behaviour is the focus. Within this orthogonal view, there is an inherent need to show the relationships and dependencies amongst the involved roles, especially when all the involved roles may potentially start at different times and end at different times.

Performance of such systems can be very difficult to analyze. Distributed systems are already a challenge to evaluate, but the problem intensifies when they are comprised of "black-box" sub-modules: software from many different (perhaps competing) vendors, usually without the source code.

While complex systems exhibit performance problems that often grow out of the system complexity, we propose they can by diagnosed by focusing on the performance of the sub-

modules. We contend that performance-oriented research must include analysis of the performance of a global system, based on the performance of its constituent sub-modules.

1.2 UML Collaborations

Due to the *cross-cutting* nature of services, there is a need to specify services in terms of their roles and cross-cutting service behaviour, then to specify the detailed dynamic behaviour of each service role and, lastly, to compose the behaviour of service roles into complete, coordinated and accurate global behaviour. UML Interaction Diagrams [62] and Message Sequence Charts [62] are commonly used for such purposes, except it is quite cumbersome to define all the scenarios. Furthermore, many a times, one does not wish or does not have enough information to model such details and would rather have the option to model at an abstract level. Also, there is a matter of realizability, i.e., finding a set of local component behaviours whose combined execution leads to the global behaviour.

A suitable candidate to model such systems is UML Collaborations. Being both structural and behavioural classifiers in UML 2 [62], Collaborations can be used to define the dynamic aspects of a system as a structure of roles with associated interaction behavior.

A UML Collaboration describes a set of cooperating roles, each performing a specialized function, which collectively accomplish some desired functionality. The relevant relationships are shown by a set of connectors that define the communication paths between the participating instances. **Roles** of a collaboration define a usage of instances, while the classifier typing these

roles specify all the required properties of these instances. A role specifies the required set of feature, a participating instance must have.

Furthermore, Collaborations can be decomposed into smaller sub-collaborations via CollaborationUse [62]. Elementary collaborations (collaborations which cannot be further decomposed) can often be specified using Interaction Diagrams. Then, the overall behaviour of a composite collaboration can be specified as a various set of sequences or a "choreography" of its sub-collaborations. Currently, UML does do not define the dynamic behaviour of the Collaboration Diagrams. A proper way to describe the joint behaviour of the sub-collaborations of a composite collaboration is needed.

1.3 Main Objective

The main objective of this research is to propose precise semantics for the dynamic behaviour of UML Collaborations in terms of partial orders of events that are performed by the different roles of the collaboration. We define the precise semantics of the usual sequence operators such as weak sequence, strict sequence, concurrency, alternatives, weak while loop and strict while loop. To do this, we model a starting event and an ending event for each role in the collaboration, where a starting event is the beginning of the first execution of that role in that collaboration and the ending event is the ending of the last execution of that role in that collaboration. We think that existing approaches to the definition of these operators are either unsuitable, or have difficulties, when modeling collaborations that have multiple independent starting points of execution (later called *starting events*) and multiple independent ending points

of executions (later called *ending events)* – simply put, not all the roles may start executions at the same time and end their executions at the same time, in a given collaboration. Complexity increases further when we consider collaborations that are decomposed into other sub-collaborations, by means of collaboration-uses.

Inspired by the Performance Evaluation and Review Technique (PERT) [18], we introduce "Collaboration Behaviour Diagrams" (CBD), a notation for the behaviour of sequencing of collaborations using the partial orders formalism. This is a modeling notation comprised of UML Activity Diagrams and a partially ordered set of starting and ending events. In CBD, the dynamic behaviour of a system is captured partially by the semantics of UML Activity Diagrams (AD). This is comprised of actions and sequencing operators to define the relationship between these actions. Each action, in general a sub-collaboration, is characterized by a partially ordered set of starting and ending events, which allows us to model situations when roles start and end their involvements in the different sub-collaborations.

With this, we can then define the semantics and compute performance of composite collaborations, composed using standard UML operators such as strict/weak sequencing, alternative, concurrency and strict/while loops based on the performance of the constituent sub-activities.

Though our work primarily caters for software systems and services, it can easily be adapted to non-computer systems with little to none modifications. Also, throughout this thesis, we use roles as compared to system components, where components are typically defined as a

physical entity and role is a logical entity, implemented by a component. Multiple roles can be implemented by a single component.

1.4 Main Contributions

The main contributions which we made as part of the research are the following:

- Introduce Collaboration Behaviour Diagram (CBD) to formalize the semantics of the dynamic behaviour of UML Collaborations with partial orders. The CBD defines the semantics for Collaborations sequenced with strict/weak sequencing, alternative, concurrency and strict/weak while loops.
- Design of a testing methodology to determine the dependencies and the delays between the starting and ending events of involved roles. Quite often, the inner details of a collaboration are not provided and the collaboration is a black-box system where there is no knowledge of the dependencies nor delays between the starting and ending events of the involved roles. This testing methodology determines these dependencies and the delays without relying on any information about the internal structure of the given activity.
- Using the CBD, we propose formulas to calculate the performance of global collaborations based on the performance of sub-collaborations. For a composite collaboration, composed of two sub-collaborations with various sequencing operators, we classify the roles according to their participation in these sub- collaborations. Based on the role classifications, we calculate the performance from the starting event of one role

to the ending event of another role (and/or the same role), based on the sequential operators and the performance of the constituent sub- collaborations. We calculate the performance for a single control flow path, and we also consider all possible control flow paths that may exist. For a given path, we consider different types of delays such as fixed, stochastic and a range of delays.

- For stochastic delays, we propose approximations for the cumulative distribution functions of Normal Distributions, as this distribution can be used to approximate other distributions due to the Central Limit Theorem.
- We calculate the delays between starting event of a role and the ending event of another role (and/or the same role), for each role in a collaboration. This gives us n^2 delays for n roles. Calculating delays becomes a very cumbersome task, for example a collaboration with 4 roles will have 16 delays, and if it is sequenced with another collaboration with another 4 different roles, this will yield $(4+4)^2=64$ delays, and that is only for a sequence of two collaborations (typically we have far more than that). To automate the calculation, we implemented the Performance Analysis Tool (PAT), a tool that calculates the performance of global collaborations based on the sequencing operators and performance of the constituent sub-collaborations. A number of examples and case studies were developed during our work to validate the tool.
- Our work relies on certain set of assumptions. These assumptions included no sharing of resources such as CPUs, database, etc. We also assumed the inputs of each collaboration were independent. Messages and their associated message delays were ignored throughout our performance analysis. We discuss the expected errors that may be

introduced due to ignoring of shared resources or possible dependencies in the case of stochastic behaviours, and how formulas can be adapted to include message delays.

1.5 Organization of the Thesis

Chapter 2 surveys a number of modeling paradigms that can be used to model the dynamic behaviour of a collaboration. We review Petri Nets, WS-BPEL, WS-BPMN, UML Activity Diagram, UML Collaborations and Partially Ordered Sets to model collaborations with multiple independent starting and ending events.

Performance concepts pertaining to analytical modeling are introduced in Chapter 3. Analytical models are classified into queuing models and stochastic models. We discuss queuing models such as Single-Server Queues, Network of Queues, and Layered Queuing Networks. For stochastic models, we discuss exponential models such as Continuous Time Markov Chains and non-exponential modeling such as Semi-Markov Processes and Generalized Semi-Markov Processes. We also examine performance extensions of some of the modeling paradigms mentioned in Chapter 2, such as Stochastic Petri Nets, Queuing Models, UML MARTE and PERT, to model these concepts.

Chapter 4 models the behaviour of a system with the Collaboration Behaviour Diagram, a formalism based on partial orders that we use to define the semantics of the dynamic behaviour of Collaborations. Also, we developed a method to determine dependencies between starting events and ending events assuming that the collaboration is a black-box system. These starting and ending events may belong to the same role and/or to different roles involved in the

collaboration. We also introduce the concept of a Nominal Execution Time Delay, the delay from a starting event of a role to an ending event of another role (and/or same role) provided that all the remaining starting events have occurred long time before. This forms the basis for the calculation of the performance of a global collaboration based on the performance of the constituent sub-collaborations and various sequencing operators.

Chapter 5 analyzes the performance of global collaborations based on the performance of the constituent sub-collaborations and the sequencing operators. We consider weak sequence, strict sequence, concurrency, alternatives, weak while loop and strict while loop. We propose formulas to calculate the delays of composite activities abstracting each of these operators.

In Chapter 6, we approximate the results for stochastic delays of Chapter 6 by Normal Distributions.

Chapter 7 discusses the assumptions that were made during our performance analysis. We discuss the impact of error that could be introduced by shared resources and dependent activities. We also discuss the existent of various messages and how one can incorporate the message delays in the performance analysis of Chapter 5.

Chapter 8 presents the Performance Analysis Tool (PAT) implemented to calculate the performance of a global collaboration. A discussion of the required input and expected output is given. We designed, developed and tested PAT in Java, using the Eclipse development environment. A Collaboration Behaviour Diagram is first modelled in the IBM Rational Software Architect (RSA) as annotated UML Activity Diagrams, which are exported in the XMI

(XML Metadata Interchange) format. This XMI model is then used as an input to PAT, where PAT analyzes the performance of the sub-collaborations and generates the performance of the global collaboration.

Chapter 9 presents the case study "Travel Management System" to illustrate the formulas from Chapter 6. The Travel Management System is modeled as a Collaboration Behaviour Diagram and the performance of the constituent sub-activities is defined. Using formulas from Chapter 6, we calculate the performance of the global collaboration. Some of the detailed results are given in Annex A and B.

Chapter 10 provides our conclusions and suggestions for future research directions.

2. MODELING

A modeling language is employed to represent information or knowledge in a structure that is defined by a set of rules and regulations. These rules and regulations define the meaning of the elements in the given structure. A modeling language can be categorized as either graphical or textual. Graphical Modeling typically uses a diagram with symbols that correspond to certain predefined concepts and lines connecting these symbols to show the relationship and/or constraints between these symbols. Meanwhile, textual modeling languages typically represent these concepts and relationships by standardized keywords along with parameters. Usually textual modeling languages are used more for computer-interpretable expressions such that there could be some automated processing done on these expressions. This can make it sometimes difficult for human readability. While graphical modeling is easier in terms of human readability, it has limited success in automated processing.

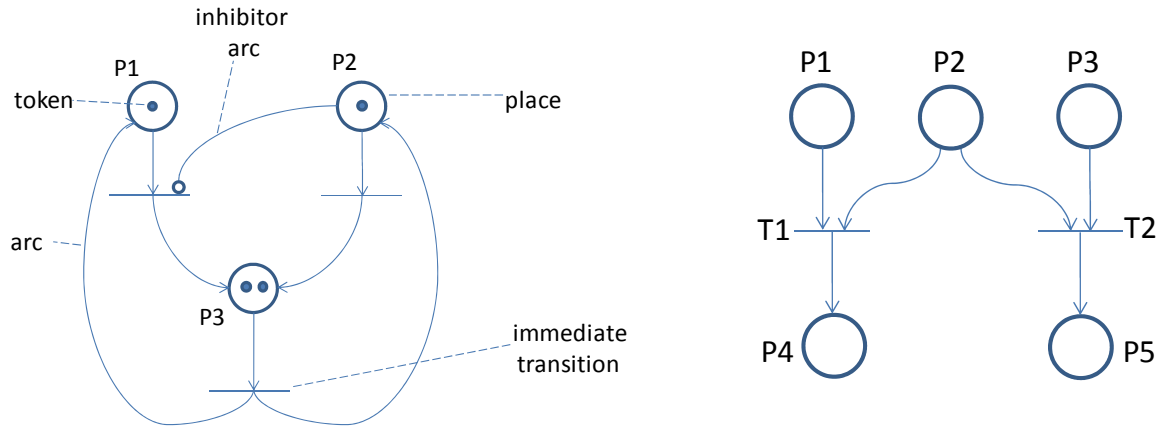
Some graphical modeling paradigms that were considered in this research include various flavors of Petri Nets (PN), Business Process Execution Language (BPEL), Business Process Model and Notation (BPMN), various diagrams of the Unified Modeling Language (UML), Queuing Models (QM), Performance Evaluation and Review Technique (PERT) and Partially Ordered Sets.

2.1 Petri Nets (PN)

Petri Nets provide formal, graphical, executable techniques for the specification and analysis of concurrent, discrete-event dynamic systems. A PN is a bipartite graph, specified by a

set of places (drawn as circles), transitions (drawn as lines) and arcs as illustrated in Figure 1a [7]. Arcs run from a place to a transition or vice versa, never between places or between transitions. An inhibitor arc imposes the precondition that the transition may only fire when the place is empty (shown as arcs with open dots). The places from/to an arc that run to/from a transition are called the input/output places of the transition. A marking of a PN is an assignment of tokens to the places of the net. A transition is enabled if there is at least one token in each of its input places and no token in its inhibitor arc—otherwise it is disabled. An enabled transition may fire by consuming a single token per place from its input places and places one token per place to each of the output places.

Two transitions are said to be in conflict when the inputs are overlapping such as shown in Figure 1b. If there is only one token in each place, then transitions T1 and T2 cannot be executed at the same time, but one can (a choice must be made).



a – Basic Petri Net [7]

b – Petri Net with Choice

Figure 1 – Petri Nets

The marking of a PN corresponds to the state of a system and firing of a transition corresponds to the occurrence of an action (or state transition). In general, some transitions are enabled while others are disabled, reflecting the fact that some actions can occur whilst others cannot. When multiple transitions are enabled at the same time, any one of them may fire, hence making execution of Petri Nets nondeterministic. If a transition is enabled, it may fire, but it does not have to. Since multiple tokens may be present anywhere in the net (even in the same place) and firing is nondeterministic, Petri Nets are well suited for modelling the concurrent behaviour of distributed systems.

2.2 Business Processes Modeling Languages

Standardized by OASIS [60], BPEL is an executable language for specifying business process behaviour based on Web Services. In BPEL, processes use web services interfaces exclusively to export and import information. The web service interactions can be differentiated in two kinds of processes: executable business processes and abstract business processes. While executable processes model the actual behaviour of a participant in a business interaction and can be executed by a BPEL engine, abstract business processes are partially specified processes that specify the mutual visible message exchange interface of each of the parties involved, without revealing the internal behaviour.

Even though there is no standard notation to graphically model BPEL, BPMN [60, 86, 88] is a popular choice by many vendors for specifying business processes in a Business Process Diagram. BPMN is a flowchart modeling technique, similar to UML Activity Diagrams [62], to

define a Business Process Diagram (BPD). It is a network of graphical objects, which are activities/tasks and the flow controls that describe the order of execution. The main purpose of BPMN was to create an easy development model while at the same time being able to handle the complexity inherent in business processes. To that end, BPD uses four categories of elements: flow objects, connecting objects, swim-lanes, and artifacts to organize the graphical aspects of the notation.

Work has been done in [24, 39, 52] to extend UML Activity Diagrams as a UML Profile for Automated Business Processes [3] to have a mapping between BPMN/BPEL and UML Activity Diagrams. This is quite useful as UML is more widely known, and has a readily understood graphical notation with a rich set of semantics for capturing key attributes of object oriented systems [52]. This transformation will enable the extensive UML experience to be applied to the maturing web services technologies.

2.3 Unified Modeling Language (UML)

UML [62] is a standardized general-purpose modeling language for specifying, constructing and documenting the artifacts of a software system. It includes a set of graphical notations to create visual models of complex software designs. UML offers 14 standard diagram types to model a software system throughout its lifecycle. These diagrams are classified as either structural or behavioural, as seen in Figure 2.

In our research, we examine UML Activity Diagrams along with UML Collaborations to model a distributed system with multiple actors.

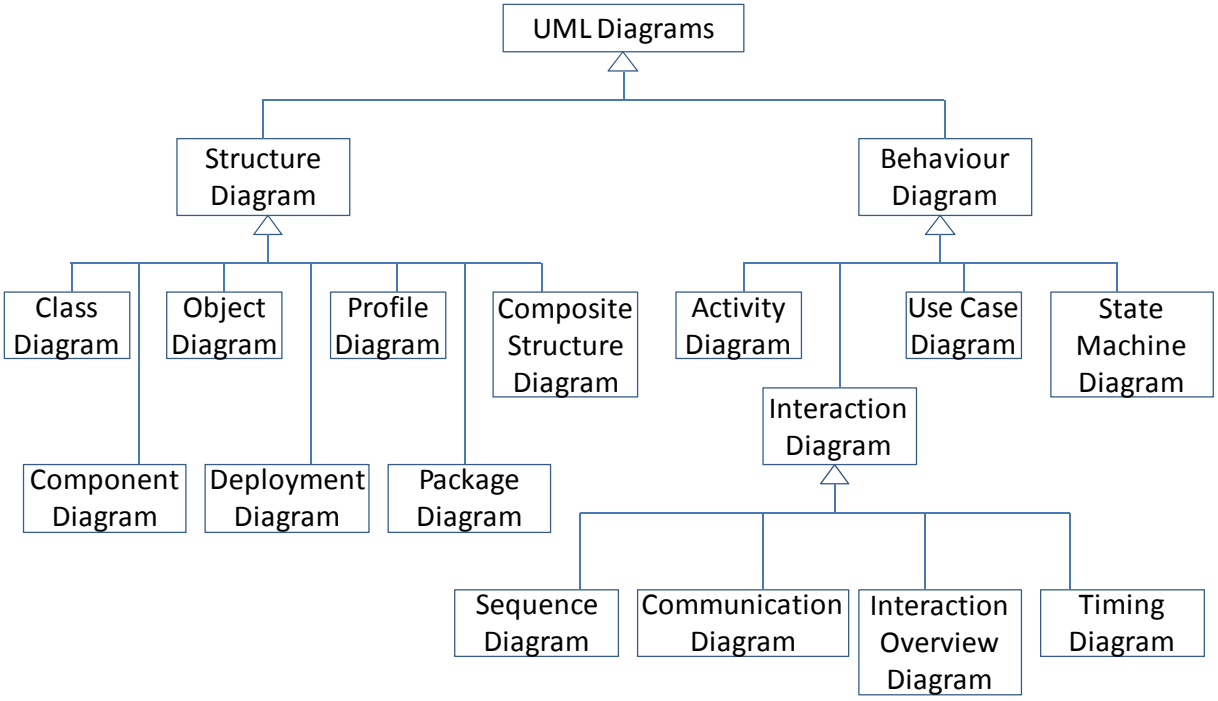


Figure 2 – UML Diagrams (adapted from [62])

2.3.1 UML Activity Diagrams (AD)

UML Activity Diagrams [62], with semantics based on Petri-Net, is a modeling formalism to describe workflows of activities and actions. Like other modeling paradigms, such as BPEL and BPMN, UML AD are typically used for business process modeling, for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule.

An AD captures the dynamic behaviour of a system. It is a flowchart to represent the flow of execution from one activity to another, composed of actions and "sequencing operators" to define the relationship between these actions. These "sequencing operators" include:

- Sequence: Allows actions to execute in a specified order
    - A sequence can be further classified as weak or strict sequencing. Though UML AD does not specify this, this distinction has been made in UML Sequence Diagrams. Strict sequencing between two activities A1 and A2 means that all sub-activities of A1 must be completed before A2 may start. In contrast, weak sequencing between A1 and A2 means that each system component locally applies sequencing to the local sub-activities of A1 and A2, that is, a component may start with sub-activities that belong to A2 as soon as it has completed all its local sub-activities that are part of A1. Note: Strict and weak sequencing operators are not associative with one another, that is the behaviour of ((A ;w B) ;s C) is not necessarily equal to (A ; w (B ;s C)), where A, B and C are collaborations and ";w" and ";s" stand for weak and strict sequencing, respectively. We give priority to strict sequence, i.e., strict sequence is evaluated first and then the weak sequence.
- Alternative: Allows a choice between several flows of actions.
- Concurrency: Allows multiple actions to be executed concurrently.
- Interruption: Allows an action (or a set of actions) to be interrupted, such that an alternative set of actions may execute.
- Loops: Allows repetition of actions to be performed.

An activity represents some transformation or processing in a modeled system. (We use *collaboration* and *activity* interchangeably through this text, as we are using activities to describe

the collaboration between several components). Such an activity may be further refined into sub-activities where the details of its behaviours could potentially be described by another UML Activity Diagram or UML Interaction Diagram. However, in most scenarios, we are not exposed to that level of detail and are only aware of the abstract description of the work being performed by a sub-activity and the roles involved.

The basic concepts for modeling an activity diagram are summarized in Table 1.

Table 1 – Elements of Activity Diagrams

Activity Model Elements	**Description**
Initial Node(s)	A filled-in circle represents the starting of a flow. There can be multiple initial nodes, each one representing a separate flow (in the cases of concurrent activities).
Final Node	A filled-in circle with a border represents the ending of all actions in an activity.
Flow Final	A circle with "X" in it, representing the end of a flow
Activity	A rounded rectangle represents a sub-activity to be executed. An activity is a collection of actions.
Flow/edge	An arc representing the flow of execution.
Fork	A black bar, with one input flow and multiple output flows, representing the beginning of concurrency of the multiple output flows.

Join	A black bar, with multiple inputs and one output flows, representing the ending of concurrency and synchronization amongst all the parallel input flows.
Condition	A text expressing a condition that must be true before continuing (ex. x > 1).
Decision	A diamond shape, representing choice between alternatives. Again this has one input, and multiple output flows but only one of those output flows gets executed.
Merge	A diamond shape, representing the merging of several alternate flows. This also has several inputs but only one output.
Partition	A swimlane, to model the separation of responsibilities
InterruptibleActivityRegion	A dashed, round-cornered rectangle drawn around the nodes contained by the region. An interrupting edge is a notation with a lightning-bolt activity edge. This is a region where any token flow through the interrupting edges will cause all the tokens and behaviours in the region to terminate.

2.3.2 UML Collaboration

A UML Collaboration Diagram [62] illustrates a collaboration between various elements (entities/roles), each assigned a certain task. Its main purpose is to explain how a group of cooperating instances achieve a desired functionality. It specifies a view of the set of cooperating instances such that it only incorporates the aspects that are relevant to the description

of the system. It describes the required links between instances that play certain **roles** in the collaboration, as well as the **features** required of the classifiers that specify the participating instances. Several collaborations may describe different projections of the same set of classifiers. Thus, a given entity may be simultaneously playing roles in multiple different collaborations, but each collaboration will only represent those aspects of that entity that are relevant to its purpose. The roles of a collaboration are played by the instances of components/classifiers. The role represents the usage of the instance, while the classifiers typing these roles specify all required properties of these instances. Thus, a collaboration identifies the properties required by instances, to participate in the collaboration, and the connectors between the roles specify the communication paths between the participating instances.

A collaboration is usually attached to an operation/classifier through a UML meta-element CollaborationUse that describes how the operation/classifier is realized by a set of cooperating entities. A collaboration is not directly instantiable. Rather, the collaboration comes about as a result of the actual cooperation between the instances that play the roles defined in the collaboration (the collaboration is a selective view of a situation).

UML defines the structural aspects of a Collaboration via Structural Classifiers. It defines the roles being used and the instances of the roles. For the behavioural aspects of Collaboration, we use the notation of UML Activity Diagram as introduced in [15]. However, we are concerned with the semantics of the dynamic behaviour that we define using partial orders. This allows us to model independent starting events that occur at different times and ending events at

different times – i.e. not all the roles start their execution at the same time and similarly for the end of executions.

As an example, we consider the collaboration of a virtual doctor's office as illustrated in Figure 3. There are 3 roles, namely the DoctorTerminal, the PatientTerminal and the ReceptionistTerminal and 5 sub-collaborations. Each of the sub-collaborations is defined separately and reused as a UML Collaboration Use. The circles and rectangles, adopted from [15] – an extension of the UML notation – respectively represent the initiating and the terminating roles of a sub-collaboration. An initiating role is a role that starts a collaboration – an initial local action in a collaboration is performed by an initiating role, and there are no other actions that precede such an action. Terminating roles are defined similarly.

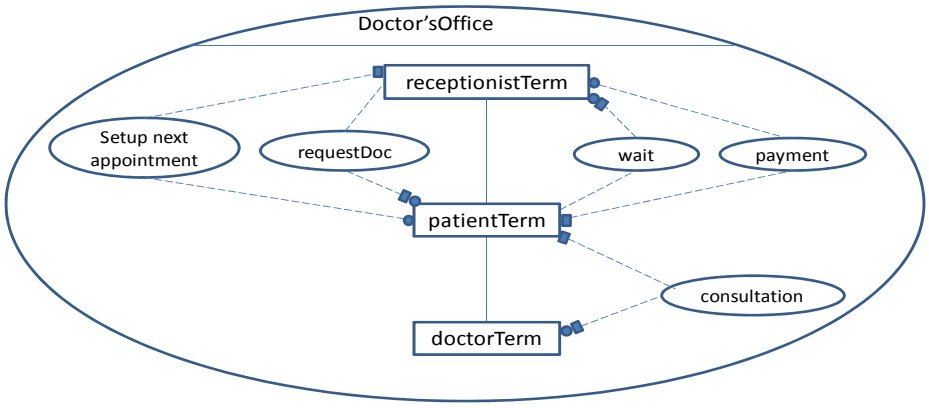


Figure 3 – Collaboration for a Doctor Office [15]

Activity Diagrams are an excellent candidate to describe the dynamic behaviour [15] of a collaboration. They provide general rules for execution orders, such as sequence, alternative, concurrency, interruptions, and also have the capability of defining partitions to separate each

role's behavior. [9] adopted the partitions from UML Activity Diagrams to separate different responsibilities inside each sub-collaboration using dashed lines, as seen in Figure 4.

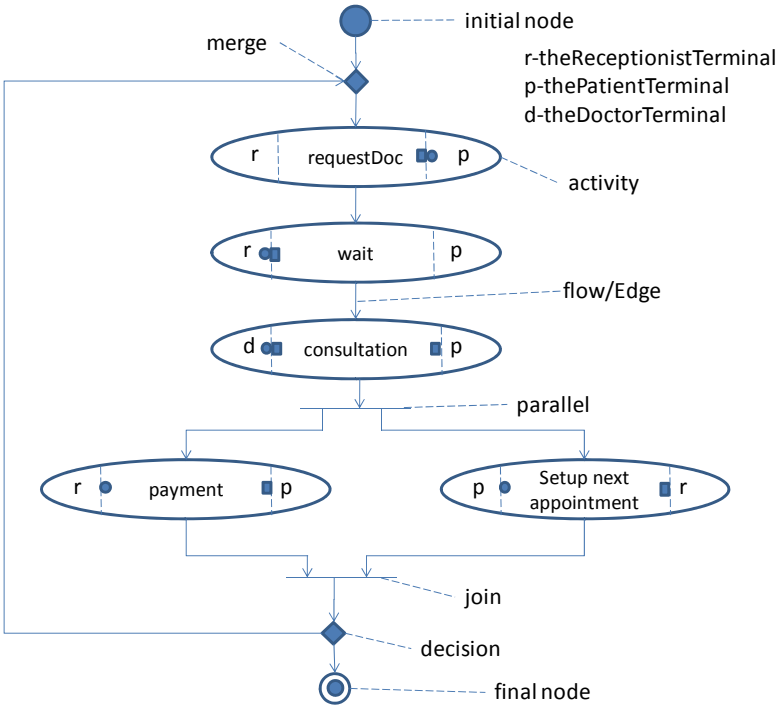


Figure 4 – Dynamic Behaviour of the DoctorOffice Collaboration [15]

The activity diagram in Figure 4 describes the global behavior of the DoctorOffice collaboration. The activity nodes are UML CallBehaviourActions that invoke the behavior associated with a collaboration type, facilitated by CollaborationUse – each of the activity nodes describes the behavior of a collaboration from Figure 3 (the same names have been kept for both the collaboration and its activity). The participating roles are indicated by the partitions, represented as dashed lines and as before, the dots and the dashes signify the initiating and the terminating roles, respectively.

2.3.2.1 Derivation of Component Specification from Global Specification

The global workflow of a system can be described by UML Activity Diagrams [62]. From this global workflow, [8] has formulated the rules to derive the individual workflows for each component/roles. A tool [45] transforms a global requirement model into a system design that identifies a certain number of distributed components. Some of the global activities involve collaboration amongst several components. The temporal constraints of the global requirement on the different activities imply a set of coordination messages between the various system components. [8] employs a set of rules to derive the behaviour of individual components including the coordination messages for the global synchronization of the activities. The global behaviour are based on the traditional sequencing operators such as weak sequencing, strict sequencing, alternatives, concurrency and interruptions. Based on this transformation, [24] has transformed a distributed system described in UML AD to BPEL with asynchronous messages.

2.4 Partially Ordered Sets (poset)

A partially ordered set (poset) is a pair $(P, \leq)$ where P is a set of elements, and $\leq$ is a binary relationship that states that for certain pairs of elements in the set, there is a precedence relationship between the two. Since this relation does not hold for all pairs of elements, it is called a partially ordered set (instead of a completely ordered set).

For a partially ordered set, the following properties must hold:

i) Reflexivity: $\forall a \in P, a \leq a$,

ii) Antisymmetry: for any a,b ∈ P, if a ≤ b and b ≤ a, then a = b

iii) Transitivity: for any a,b,c ∈ P, if a ≤ b and b ≤ c, then a ≤ c

We use the notation, $i \rightarrow$ o, to specify $(i,o) \in P$ which means that i precedes o in the partial order. The elements *i* and *o* of a poset (*P*, "→") are called comparable if either $i \rightarrow o$ or $o \rightarrow i$, otherwise they are incomparable.

For example, suppose there is a set A = {a, b, c} and P contains all subsets of A, as shown in the Hasse diagram of Figure 5. The Hasse Diagram models the elements and the predecessor relationship amongst pairs of elements. The element {∅} is the predecessor to all of the other elements. Elements {a}, {b}, and {c} succeed {∅} but there is no relationship defined amongst these elements, {a}, {b} and {c}. Similarly {a,c} succeeds element {a} and element {c}, but again nothing can be said about the order between {b} and {a,c} and so on. The diagram of Figure 5a is incomplete as it does not show all the dependencies. Because of transitivity and the relationships {∅} → {a}, {a} → {a,c} and {a,c} → {a,b,c}, we can conclude that {∅} → {a,b,c} must also exist as shown by the thick arrow in Figure 5b. Hence additional relationships can be defined using the transitive property. This is a very important property, as quite often a system is analyzed at an abstract level and hence abstract dependencies are required.

A partial ordering can be defined by a Directed Acyclic Graph (DAG). This can be accomplished by allowing the set of objects be the vertices of the DAG, and defining x ≤ y to be

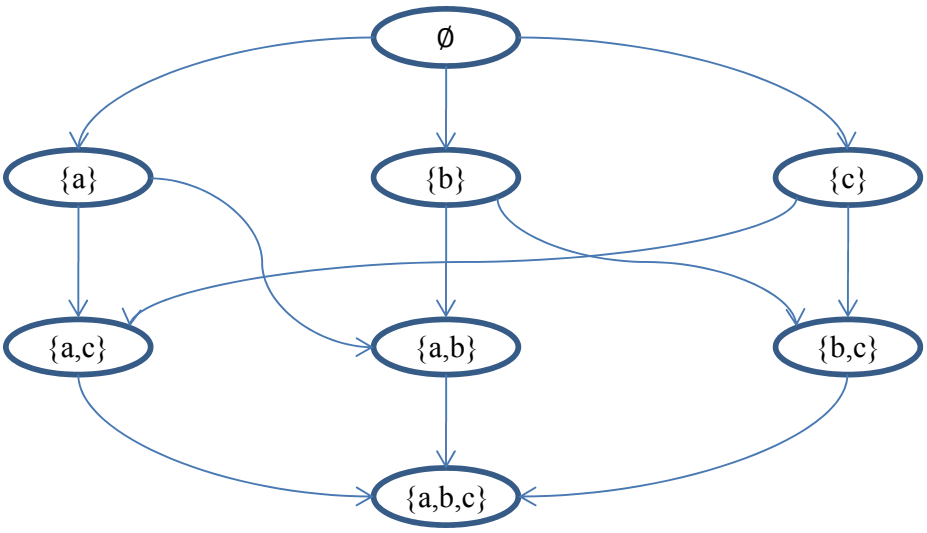


Figure 5a – Example of poset adapted from [20]

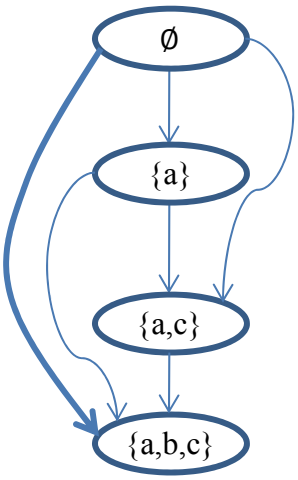


Figure 5b – Transitivity

true, for any two vertices x and y, whenever there exists a directed path from vertex x to vertex y – i.e. whenever y is reachable from x. With these definitions, a topological ordering of the DAG is the same as a linear extension of the partial order. Alternatively, any partially ordered set may be defined as the reachability relation in a DAG. One can define a DAG that has a vertex for each element in the poset, and an edge $x \rightarrow y$ for each pair of elements for which $x \leq y$ holds. Using these methods, one can use topological ordering algorithms to find linear extensions of partial orders.

Events may occur in sequence, concurrently or alternatively. We describe the partial order representations for these relationships.

2.4.1 Modeling Sequential Events

A set of events may have a dependency on other events, and hence cannot occur until all the required dependent events have occurred. Such a situation is shown in Figure 6a, where event e_4, is shown to be dependent on e_1 and e_3, while e_2 is only dependent on e_1. Events e_2 and e_4 are incomparable.

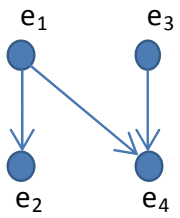


Figure 6a – Sequential Events with Partial Orders

2.4.2 Modeling Alternatives

Having an operator to model alternatives is an essential concept in UML. As there is no standardized way of modeling alternative paths with partial orders, we use the choice symbol from UCM [13] to model alternatives and the merge. As can be seen in Figure 6b, the decision is modeled as a rectangular box with one incoming edge and multiple outgoing edges. Merging of the alternative paths is modeled again as a rectangular box but with multiple incoming edges and one outgoing edge.

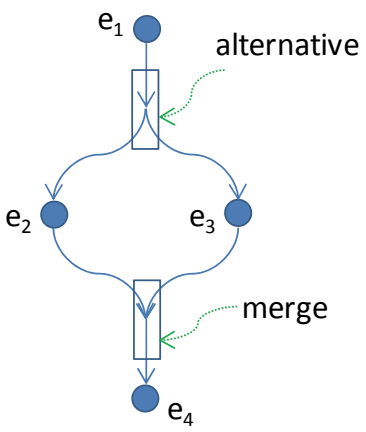


Figure 6b – Modeling a path with an Alternative in Partial Orders

2.4.3 Modeling Concurrency with Partial Orders

We can model concurrency as illustrated in Figure 6c. There is a single event e_1 which leads to two concurrent events e_2 and e_3. Events e_2 and e_3 can only occur after e_1 has occurred and hence the shown dependencies. Events e_2 and e_3 are followed by event e_4, where for event e_4 to occur, both events e_2 and e_3 must have occurred.

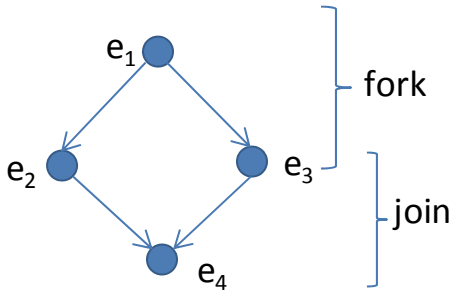


Figure 6c – Representation of Concurrency with Partial Orders

2.5 Chapter Summary

In this chapter, we reviewed various paradigms such as Petri Nets, WS-BPEL, WS-BPMN, UML Activity Diagram, UML Collaborations and Partially Ordered Sets to model the dynamic behaviour of collaborations with multiple independent starting and ending events. Even though each of these paradigms are quite powerful to model behaviours of distributed systems, they are unable to model behaviour of collaborations with multiple independent starting and ending events, where a collaboration could be decomposed into sub-collaborations. This is further discussed in Section 3.6.

3. PERFORMANCE MODELING

There are several options to evaluate performance of a system or a component during the development process. Some of these include evaluating a prototype, using a discrete event simulation (DES), or constructing analytical models.

Constructing a prototype is not necessarily always realistic during an implementation phase of the development process. It is sometimes quite difficult to assess the performance or dependability of a given system, for example it would be difficult to see the effects of performance of thousands of users interacting in a given system.

Simulation is more popular in the research domain. There are many software tools / products that can assist in the construction of these simulation models. However, this is quite expensive as the tools require writing and debugging complex programs, which may take large amount of computation resources to complete. Additionally, added expenses can include parameterizing as a highly detailed model requires a large number of parameters.

In contrast, analytical modeling is an economical alternative to simulations. This is the main focus of our research. Analytical modeling is based on the construction and analysis of a model, where a model is an abstract representation used to capture the essential characteristics of a system such that the performance of the model can be reproduced. A model should include sufficient information describing behaviour of the actual system – this could include various characteristics such as performance, fault-tolerance, resource contentions, concurrency and synchronization.

Analytic models can be classified in the following two categories:

i) Non-Stochastic Models: non-stochastic models typically describe a deterministic system, where the behaviour of a model always produces the same output for a given set of inputs. Amongst the more popular modeling paradigms are queuing models such as Layered Queuing Networks, which typically study queue lengths and waiting times to analyze resource utilization and contention.

ii) Stochastic Models: stochastic models are used for systems where the behaviour of a system is defined by a probability distribution.

Conceptually, stochastic modeling can be further refined into exponential models (Continuous Time Markov Chains (CTMC)) and non-exponential models (Semi-Markov Processes (SMP) and Generalized Semi-Markov Processes (GSMP)). Because of the exponentially distributed times and the resultant memoryless property of CTMC, CTMCs can be easily analyzed through straightforward application of numerical analysis. But for more generic cases, SMPs and GSMPs are used for modeling where an exponential distribution is not sufficient, such as for modeling a range of delays or modeling fixed delays for hard real-time systems.

In this chapter, we begin with the study of CTMC, SMP and GSMP. We will then introduce various modeling paradigms such as Queuing Models, Timed Petri Nets, Stochastic Petri Nets, Performance Evaluation Review Technique (PERT) and the UML profile “Modeling and Analysis of Real Time Embedded System” (MARTE).

3.1 Stochastic Models

A stochastic model predicts a set of possible outcomes weighted by their likelihood, or probabilities. It can be described by a state automata whose transitions are triggered by the occurrence of stochastically timed events associated with the occupied state. This means that even if the initial condition (or starting point) is known, there are many possibilities the process might go, but some paths may be more probable than others.

The stochastic models that are most commonly used in performance and dependability analysis are Continuous-Time Markov Chains (CTMC), Semi-Markov Processes (SMP) and Generalized Semi-Markov Processes (GSMP). These models vary by the set of instants in a process's life that satisfy the Markov property. A state of a stochastic model has the Markov property if the conditional probability distribution for the next state of the model, given the present state and the past, depends only upon the present state; that is, the past is irrelevant because it does not matter how and when the current state was reached. The Markov property holds at all instants of the process life for a CTMC. For SMPs, the property holds at the instants of the state changes only, and for GSMP, the property may never hold. Since the Markovian property is true at all instants, CTMC are models where durations can be represented by exponentially distributed distributions. In contrast, SMPs and GSMPs can represent systems with arbitrarily distributed durations.

Only continuous time models will be considered in this thesis, i.e., inter-event time, t, takes its value from the set of positive real numbers. A model is described as irreducible (that is,

strictly connected) if all states can be reached from all other states, by following transitions of the process.

3.1.1 Exponential Models

The exponential distribution (also known as negative exponential distribution) describes the time between events in a Poisson process – i.e. a process in which events occur continuously and independently at a constant average rate. A Poisson process is a counting process in which the inter-arrival time of successive jumps are independently and identically distributed exponential random variables.

3.1.1.1 Continuous Time Markov Chain (CTMC)

In a CTMC, each state in the automaton is associated with several Poisson processes. A transition between any two states of the automaton is governed by the jump of one of the Poisson processes associated with the occupied state. These Poisson processes compete to trigger the next transition. The distribution of the time spent in a state of a CTMC, also known as sojourn time in the state, is actually exponentially distributed.

A Continuous-Time Markov chain with a small state space is commonly represented as a state transition diagram. Each state is represented by a node and the transitions between various states by arcs. The arcs are annotated with the parameters of the exponential distributions governing these transitions: for instance, q_{ij} is the transition rate for a transition from state i to state j.

3.1.2 Non-Exponential Models

Due to their memoryless property, CTMC can be easily analyzed with straightforward application of numerical analysis, making the model quite popular. However, an exponential distribution is not always realistic. There are scenarios where other distributions would be more appropriate – for example, if only the minimum and maximum of some quantity is known, then the uniform distribution would be preferred. File transmissions and files sizes tend to lean towards heavy-tail distributions, etc.

If the transition delay is of non-exponential distribution, the Markov property need not hold anymore—the future behaviour of the process might depend on a distribution that was started in the past and has not triggered a transition yet. A Semi-Markov Process (SMP) is a stochastic process which allows non-exponential transitions and has the Markov property at the time of any state change. In a SMP, all distributions that govern transitions are initialized every time we enter a state; as a result, the lifespan of distributions from past states is not carried over. Semi-Markov Processes (SMP) is a subset of Generalized Semi-Markov Processes (GSMP), where the distributions are not necessarily initialized at each transition.

3.1.2.1 Generalized Semi-Markov Process (GSMP) and Semi-Markov Processes (SMP)

A GSMP is a state automaton where transitions are triggered by the occurrence of stochastically timed events. A set of active events, A(s), is associated to each state, which can cause an outgoing transition from that state. Each of these events has their own distribution for determining the next state. At each transition to a state s, a set K(s) of new events is scheduled.

The sojourn time (according to the random distribution) of each event is determined by their individual clocks, and hence one clock per event. If an event e' occurs causing a transition from state s to state s', and if another event e was active in state s, then either event e is associated with the new state and its clocks continues to run or e is not associated with the next state, resulting in its lifetime being discarded (or aborted) and is considered inactive. A transition between two states is labelled by an event e and a set of events E that are aborted, represented as $s \xrightarrow{e}_{E} s'$. This represents the clock of event e expiring, in s and the process aborts the events in E and moves to state s'. As a result, the active events in state s' would be $A(s') = A(s) - (e \cup E) + K(s')$. The events in the set $A(s) - (e \cup E)$ keep their residual lifetime, while events in K(s') are initialized according to their distributions.

SMPs are a subset of GSMPs where $K(s) = A(s)$ for all s, meaning that all the active events in a state are initialized once that state is reached. Due to this, SMPs satisfy the Markov property at the time of state change.

3.1.3 GSMPs versus CTMCs and SMP - Comparison

To fully understand these various modeling paradigms, it is quite useful to discuss the similarities and differences between GSMP, CTMC and SMP.

A CTMC is a GSMP in which all the clocks are governed by a negative exponential distribution, resulting in a Markov Property at all instants of the process life. The Markov property means that the probability of making a transition to a next state only depends on the current state and not on previous states ("absence of state memory"). A CTMC also implies that

the probabilities of taking next transitions do not depend on the amount of time spent in a current state ("absence of age memory"). The residence time of a state is exponentially distributed with a rate that equals the sum of the rates of its outgoing transitions.

A SMP is a GSMP in which all clocks are initialised on each state change, resulting in no residual time from the previous state to carry over. For an SMP, the state residence times is determined by the distributions of the different possible transitions (which may have different distributions, each). In a GSMP, the residence time of a state is determined implicitly by the distributions of the set of active events in a state. As a result, the state residence times in a GSMP may be history-dependent. This phenomenon is the essential difference with SMPs, where state residence times are governed by an a priori, fixed random variable.

Also as exponential distributions have memoryless behaviour, it does not make a difference between initializing clocks on each state change or not. As a result, each CTMC is an SMP but an SMP is not a CTMC, in general. A SMP possesses the Markov property, but does not satisfy the "absence of age memory" principle: probabilities of next transitions are dependent on the amount of time spent in the current state.

3.1.4 Application of SMP and GSMP

Since Markov Chains represent only activities with an exponentially distributed duration (distributions with memoryless property), the only candidates to represent systems with generally distributed durations are SMP and GSMP.

SMPs are processes where the durations are initialized every time the state is entered. Activities in series and alternatives in Figure 6a and c are good examples of SMPs, where the

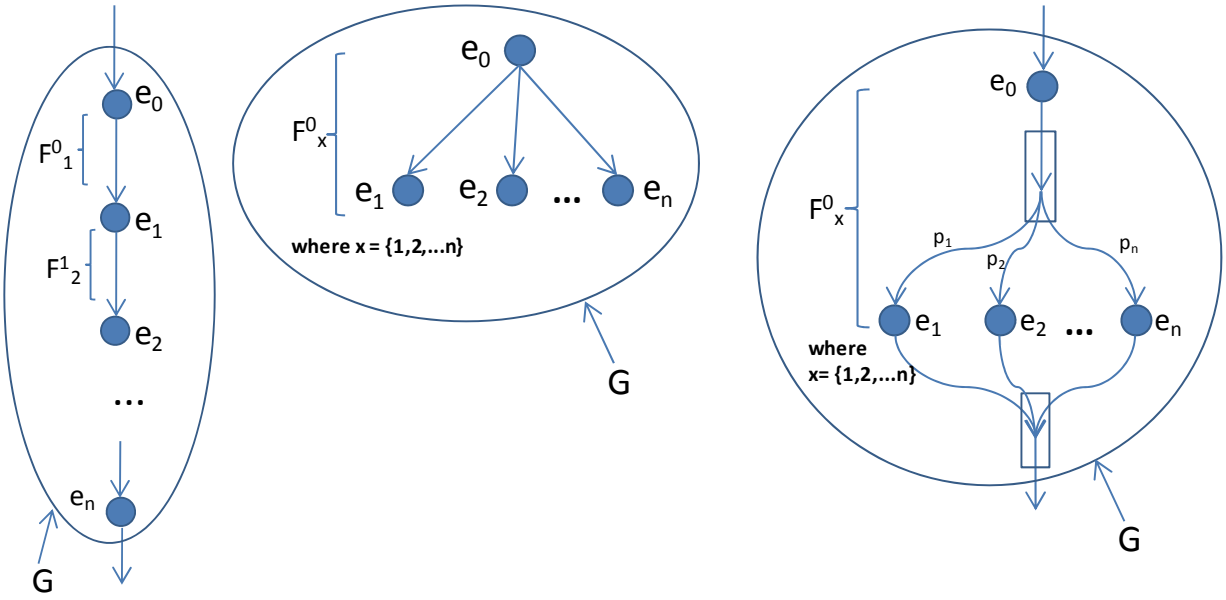
transition to the next state is dependent only on the current state and not on the past states (Markov Property). The duration of the transition to the next state is initialized every time a state is entered, and the sojourn time of any remaining *untriggered* transitions of the previous state is discarded.

In contrast, activities in parallel imply that all the activities are executing concurrently. Activities may stop their execution as soon as the first activity has completed its task or all the activities may run to completion. However, let us consider the example of a parallel search for an item in a distributed database, where the earliest concurrent search process to finish, will terminate the overall search. We would be interested in the time it would take for the earliest search to finish. This would be modeled as a SMP as transitions where the remaining time are discarded. However, if we allow all the activities to run to completion, then this would be modeled as a GSMP, as none of the times of any transitions is discarded.

The duration of an activity can be modeled by a distribution. If the duration has a fixed value, it may be modeled by a delta distribution (representing a deterministic duration). A cumulative distribution function (CDF) of a real valued random variable X is $F_X(x) = P(X \leq x)$, where $P(X \leq x)$ is the probability that the random variable X takes on a value less than or equal to x.

Let us consider the various orderings of events e_0, e_1, ...e_n in Figure 7. Let us abstract the sequence of events e_0, e_1,…,e_n by a composite activity G, where composite activity G abstracts events in series in Figure 7a, concurrent events in Figure 7b and alternative set of events in Figure 7c. We assume there exists a stochastic delay, $F^x_y(t)$, between events e_x and e_y as shown in Figure 7a-c, characterized by a cumulative distribution function (CDF) $F^x_y(t)$. We assume

these delays are statistically independent. If the distributions of the delays between the events are considered to be a delta distribution, then these delays have a deterministic duration, and we have the situation of "Fixed Delays". In the following, we examine fixed delays and stochastic delays for the abstract activity G for the series, alternative and concurrent operators.



a – series b – concurrency c – alternative

Figure 7 – Various orderings of Events

3.1.4.1 Series

The events in Figure 7a are ordered in series, i.e. there is only a single event succeeding each other and they are processed in a straightforward sequence.

Stochastic Delays:

If the delays between the events of Figure 7a are stochastic delays, then the CDF F_G for the abstract activity G, is defined by [75] as:

$$F_G(t) = \bigotimes_{i=0}^{n-1} F^{i}_{i+1}(t), \quad (1)$$

where ⊗stands for convolution, defined as:

$F_j(t) \otimes F_k(t) = \int_0^t F_k(t-x)\, dF_j(x)$ -- note the order of the activities do not matter

Fixed Delays:

If the distributions of the delays between the events are considered to be a delta distribution and hence a fixed delay, then the delay for the abstract activity G can be calculated by:

$$F_G = \sum_{i=0}^{n-1} F^{i}_{i+1} \quad (2)$$

3.1.4.2 Concurrency

Stochastic Delays:

Now, suppose if we wish to calculate the overall duration for the SMP case of concurrency, we would be interested in the time it would take for the earliest event amongst e_1, e_2,…e_n after event e_0 to occur as shown in Figure 7b. A scenario with the earliest or "minimum" CDF of a global activity G composed of parallel events e_i, can be modeled by [75] as:

Completion of the Earliest Event: $$_{min}F_G(t) = 1 - \prod_{i=1}^{n}(1 - F^{0}_{i}(t)\ ) \quad (3)$$

Next, we examine the GSMP case of concurrency, where there are parallel events, e_i, composing the global activity G again, but the completion of this global activity G requires all of the events e_i to occur. This can be accomplished by calculating the delay between events with the maximum time delay, [75]:

Completion of the Last Event: $${}_{max}F_G(t)= \prod_{i=1}^{n} F_i^0(t) \quad (4)$$

Fixed Delays:

If we consider the delays between the events as a delta distribution, and if we consider a range of delays for the deterministic duration of the abstract activity G, then the range of delays can be defined as:

Completion of the Earliest Event: $${}_{min}F_G = \min_{i=1\ldots n}(F_i^0) \quad (5)$$

Completion of the Last Event: $${}_{max}F_G = \max_{i=1\ldots n}(F_i^0) \quad (6)$$

3.1.4.3 Alternatives

Alternative execution is when there is a decision to be made amongst multiple successor events and only one of the successor events occurs as shown in Figure 7c.

Stochastic Delays:

To obtain a distribution delay, each path is assigned a probability value p_i leading to alternative paths i. If we examine the alternative case, a SMP, such as in Figure 7c, then the overall distribution of G is defined by [75]:

$F_G(t) = \sum_{i=1}^{n} p_i F_i^0(t)$ (7)

where p_i is the probability for event e_i to occur.

Fixed Delays:

The fixed delay of the abstract activity G is the delay of the particular single control flow path selected to be executed:

$F_G = F_i^0$ (8)

where e_i is the event occurs and $1 \leq i \leq n$

If we consider the range of delays, then we need to consider all the control flow paths which would lead to the greatest and smallest delay:

Minimum delay: $_{min}F_G = min_{i=1\ldots n}(F_i^0)$ (9)

Maximum delay: $_{max}F_G = max_{i=1\ldots n}(F_i^0)$ (10)

These formulae can be used to determine the distribution of a global scenario given the distribution function of between individual events. They will be the basis of our work in Chapter 5.

3.2 Non-Stochastic Models

3.2.1 Queuing Models

A queuing model consists of one or more servers that provide service to arriving customers. Arriving customers, who find all servers busy, generally join a queue, waiting to be processed by the servers, and thus the name "queuing model". A queuing model is typically

used to approximate a real life queuing situation or a system such as bank-teller service, or customers waiting at the checkout of a grocery store. With this model, the queuing behaviour can be analysed mathematically and useful statistics can be realized. Some of these statistics include the average number of customers in a given queue, the average time spent in a queue, the statistical distribution of those numbers, the probability of a queue being full (or empty), and the probability of finding the system in a particular state. Analyses of such statistics help us determine the source of issues related with queues. As well, we can also predict the impact, if there is any, of any design changes in a given system and react accordingly.

There are several types of queuing models for a modeller to use as the need arises. Amongst them, some of the more popular ones are Single Server Queues, Network of Queues, and Layered Queuing Network (LQN).

3.2.1.1 Single Server Queue (SSQ)

In a single server queue model, there is a single server along with a queue of infinite capacity. Successive customers with random service requirements arrive at random interval times to be serviced by the server. If the server is busy, the customers queue up and upon the server's availability, based on a predefined principle (i.e. FIFO, LIFO etc), the customers are processed by the server uninterrupted and depart the system upon completion.

3.2.1.2 Network of Queue (NoQ)

The term 'network of queues' describes a situation where the input to one queue is the output from one or more servers. This is true in many situations in the area of

telecommunications or computer organization. If we view an assembly line, where computer parts are assembled, and these parts move from one assembly station to another sequentially, then the system can be modelled as a queuing network. Each assembly station is modelled as a separate service centre with its queue. Parts moving between the stations correspond to the users in the system. Each part may move from one assembly station to another in the system, the pattern in which these parts move is usually predefined. Each assembly station has a single queue, however, its characteristics (arrival, service time, etc) are not independent of the other assembly stations within the system. Such a situation is called a queuing network.

3.2.1.3 Layered Queuing Networks (LQN)

LQN models are an effective modeling paradigm to model complex resources that are held in groups when a program executes [90, 91]. This modeling paradigm is used to model layered service systems such as client-server, web applications, etc, where upper-layer servers include the use of lower layer servers in their service time. Performance measures such as throughput bounds, mean delay, processor utilizations and queuing delays are estimated with the help of this methodology. LQN describes a system by the set of resources that are employed by its operations. One or more resources are required by each operation for which the model defines a resource and an architectural context. The architecture context is a software object to execute the operation, and the resource context is a set of software and hardware entities required by the operation.

3.2.2 Queuing Models: Pros and Cons

Queuing models can be constructed and evaluated relatively easily. However, some aspects of the computer and communication systems cannot be represented by queuing networks. It is quite difficult to represent systems in which there is internal concurrency or systems in which more than one resource is acquired using queuing models.

3.3 Petri Nets

There are many extensions of Petri Nets to suit various purposes. For performance modeling, some of these extensions include Timed Petri Nets (TPN) and Stochastic Petri Nets (SPN).

3.3.1 Timed Petri Nets (TPN) [27, 71]

As powerful as the Petri Net formalism is, it is not sufficient for modeling performance evaluation where quantitative measures such as throughput, response time, and average marking are modeled. The idea of adding time to PNs is to introduce the temporal constraints on its elements such as "a transition must fire within 20 time units" or "a token must remain in a certain place for at least 5 time units". So, temporal constraints are added to transitions, places, and arcs. Most, if not all, possible approaches have been studied in the literature, adding time constraints to places only, transitions only, arcs only or a combination of any of these. For illustration purposes, temporal constraints on transition will be reviewed.

There is a lower bound d and an upper bound D to a transition. If transition t is associated with constraint [d, D] (with $d \leq D$), then, after t is enabled, it must fire no less than d and no more than D time units after it is enabled (unless it is disabled before). As can be seen in Figure 8, transition t, once enabled, can fire not before 2 time units and not after 3 time units. If a token arrives in P1 at time $T_1 = 4$, P2 at time $T_2 = 6$, and P3 at time $T_3 = 2$, then transition t fires non-deterministically between time $T_{min} = 8$ and $T_{max} = 9$.

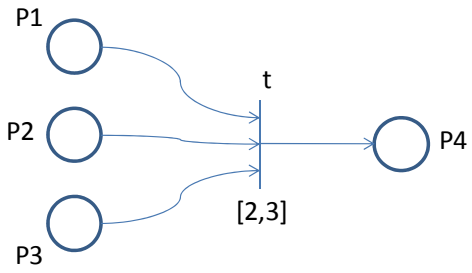


Figure 8 – Timed Petri Nets

3.3.2 Stochastic Petri Nets (SPN) [1, 7, 84]

SPNs were introduced in 1980 [82] as a formalism to describe Discrete Event Dynamic Systems (DEDS) whose behaviour could be represented by Continuous-Time Markov Chains. Stochastic Petri Nets (SPNs) are Timed Petri Nets with atomic firing in which the firing delays are specified by exponentially distributed random variables, i.e. each transition t_i is associated with a random firing delay whose probability density function is a negative exponential with rate w_i. SPN consists of a race execution policy, i.e. when multiple transitions are simultaneously enabled, the race policy selects the transition with the statistically minimum delay to fire.

The exponential distribution for the firing delay is quite useful since a SPN, where transitions have an exponentially distributed delay, can be mapped onto Continuous-Time Markov Chains. Due to the memoryless property of the exponential distribution of firing delays, SPN systems are isomorphic to continuous time Markov chains (CTMCs).

3.4 Performance Evaluation and Review Technique (PERT)

Introduced in 1950s by the American Navy, PERT is a network-based methodology that consists of planning and scheduling the many inter-related tasks in a large and complex project [18], especially the time needed to complete each task and to identify the minimum time needed to complete the total project [69]. PERT uses a network representation to illustrate precedence and parallel relationships amongst activities in a given project.

In PERT, an activity in a project is a task that must be performed and an event is a milestone which indicates the completion of one or more activities. The PERT formalism comprises the following concepts:

PERT events (nodes): Nodes are events or points in time. This marks the start or the completion of one or more activities. It consumes no time nor any resources. An event only "occurs" after all the activities leading to this event have completed. In a PERT network, there is single starting node which has only outflow arcs and a single ending node, which has only incoming arcs.

PERT activities (arcs): Directed arcs represent activities. An arc represents the execution of a task that has a time delay associated and requires resources. The activities leaving a node cannot begin until all the activities (arcs) entering the node have completed their executions.

This leads to having no cycles in a PERT network, as if an outflow activity cannot begin until all of its inflow activities have completed, a cycle would mean the system can never continue from that point onward. This also means that alternative flows cannot be modeled in PERT.

The PERT development process comprises the following steps [18, 69]:

i) Identify the specific activities and milestones

ii) Determine the proper sequence of the activities

iii) Construct a network diagram

iv) Estimate the time required for each activity

v) Determine the critical path

vi) Update the PERT chart as the project progresses.

Amongst acquiring other statistic, the PERT model helps answer the following questions:

i) Shortest time for completion of a project

ii) What activities are on the critical path – this is the longest possible continuous path from an initial event to the terminal (last) event. Any additional delays along the critical path will delay the reaching of the terminal event by at least the same amount.

3.4.1 Pros and Cons of PERT

PERT, as mentioned earlier, cannot model a system with loops/cycles nor model any alternatives.

The network chart can easily grow quite large as the number of activities and the dependencies increase, making the process unmanageable very quickly.

As PERT is based on original estimates of individual activities, in real life, these time estimates may not hold true due to certain uncertainties. This causes PERT to provide inaccurate information about the project completion time. A solution to this problem is to add safety robustness in order to absorb any disruptions.

PERT also provides the activities on critical path that can directly impact the completion time. Also, it identifies activities that have slack time and therefore could provide resources to activities on the critical path in order to improve the overall process. This is quite useful, as illustrated in Chapter 9.

PERT can explicitly define the dependencies between the activities such that one knows what dependencies need to be fulfilled for the complete execution of the behaviour. With these dependencies, it can also calculate the project completion time. These are very powerful properties of PERT which we also use in our research.

3.5 UML

Many extensions of UML are represented by profiles. Profiling is a mechanism in UML that allows a modeller to extend certain aspects of the UML notations for particular domains (ex. health care, financial) and platforms (J2EE, .NET). These profiles are defined using stereotypes, tag definitions, and constraints that are then applied to the UML meta-model.

Seeing the need for an extension in the area of real-time and embedded software systems, OMG adopted a UML profile called “UML Profile for Schedulability, Performance and Time (SPT)”. However, experience with SPT revealed shortcomings in terms of expressive power and

flexibility. With the emergence of the new UML 2.0, it became necessary to upgrade SPT and a request for proposal (RFP) was issued seeking out a new profile, resulting in "Modeling and Analysis of Real Time Embedded Systems" (MARTE).

MARTE extends UML to support Real Time and Embedded Systems (RTES). This extension provides support for specification, design and verification/validation stages as well as support to model component-based architectures, and a wide variety of different computation paradigms (including asynchronous, synchronous, and time). The advantages of using MARTE are the following [53]:

- MARTE provides a common way of modeling both hardware and software features of a RTES to enable better communication between developers,
- MARTE enables interoperability between various development tools used for system specification, design, implementation, verification, etc,
- MARTE promotes the development of models which may be used to make quantitative predictions of both software and hardware characteristics in regards to RTES.

MARTE is defined in three parts: MARTE Foundations, MARTE Design Model and MARTE Analysis Model. The Analysis Model is quite applicable to our work. In this part, Generic Quantitative Analysis Modeling (GQAM) is described. GQAM includes analysis based on software behavior such as performance and schedulability as well as power, memory, reliability, availability and security. MARTE further refines GQAM into Performance Analysis Modeling (PAM) and Schedulability Analysis Modeling (SAM). PAM is used to annotate the

existing UML meta-models such as activities, actions and flows/edges to contain performance parameters such as duration delays, number of repetitions counts, etc.

3.6 What's Missing

We have surveyed various modeling paradigms (performance and non-performance models) in an attempt to model a distributed collaboration with multiple independent starting (inputs) and ending (outputs) events at different times, where a collaboration may be decomposed into further sub-collaborations. UML Diagrams (Activity Diagrams and Collaboration Diagrams) and Business Process Modeling Languages allow modeling of multiple starting and ending events. However, they inherently assume all the starting events occur at the same time. Petri Nets allows modeling of multiple starting events at different times. However, compositionality of activities involving weak sequences is not possible with Petri Nets. In Petri Nets, when a transition is triggered, a token is sent to all the *receiver* places. This does not allow composition of transitions to model weak sequencing. PERT does not allow modeling of alternatives and loops.

3.7 Summary

In this chapter, we classified models to have stochastic and non-stochastic behaviour. For models with stochastic behaviour, we further categorized models into exponential models represented by CTMCs and non-exponential models, represented by SMPs, and GSMPs. To model non-stochastic behaviour, we surveyed various flavours of queuing models, Petri Nets,

PERT and UML MARTE. These models were examined in an attempt to model the performance of behaviours of collaborations with multiple independent starting and ending events

4. MODELING DISTRIBUTED COLLABORATION SERVICES

During the development of a distributed system, there is often a requirement to model behaviour, both from a global system perspective and a local/component-wise perspective. In many cases, several components in a given system collaborate with each other to produce a global behavior. This need has been identified by several authors such as [8] and [14] and has been referred to as the "cross-cutting" nature of services. Our understanding of "service" is an identified functionality intending to establish some desired goals/effects amongst collaborating entities. Often, each component takes part in several different services, so in general, the behavior of services is composed from partial component behaviours, while component behaviours are composed from partial service behaviours. The global view is necessary to understand, specify, and analyze the collaborative behaviour of services provided by a system, while the local perspective is needed to precisely design the system and its components.

In the past, it has been common to model the local view of distributed reactive systems as a set of loosely coupled components modeled as communicating state machines, using languages such as SDL and lately UML state diagrams. This has improved quality and modularity, mainly by providing means to define a complex, reactive behaviour precisely in a way understandable by humans and suitable for formal analysis and automatic code generation. However, with this approach, even though the individual components are defined individually and precisely, the behaviour of the service, in which the components participate, gets fragmented.

To express the service behaviour as precisely and completely as possible, various modeling formalisms such as MSCs, and UML Interaction Diagrams have been used. However, as discussed in [14], recognized issues such as incompleteness, realizability, and compositionality, are associated with global behaviours and their relationships to local behaviours. To overcome these shortcomings, we use UML 2 Collaborations to describe the global perspective of distributed services, but only the structural aspects – not the dynamic behaviour. UML Collaborations depict the structures of collaborating elements, called roles, each executing a specific function, which collectively accomplish a common goal.

UML Collaborations allow the specification of compositions of sub-services. Collaborations defining other services can be referenced by other Collaborations through *Collaboration Use*. A *Collaboration Use* specifies how the roles of the referenced collaborations are bound by the containing Collaboration. In effect, Collaborations referencing other collaborations can be thought of as composite collaborations, while collaborations being referenced are sub-collaborations. Hence, Collaborations directly allow service modeling and service composition.

When collaborations are refined structurally, it is often the case that the behaviour of the resulting sub-collaboration behaves trivial enough to be completely defined using UML Interaction Diagrams or similar modeling formalism. However, the question of how to define the global behaviour of composite collaborations in terms of sub-collaboration behaviours (i.e., the ordering of the execution of the sub-collaborations) along with the associated performance remains.

To answer this, we explore Collaboration Behaviour Diagrams. First, we specify our understanding of collaboration, actors and events.

4.1 Collaborations, Actors, and Events

A **collaboration** specifies behaviour of a given distributed system. This includes, but is not limited to, specification of interaction between a set of actors, execution of primitive functions such as arithmetic functions, invocation of other collaborations (and/or itself), and manipulation of objects such as reading/writing of attributes. The actions are sequenced using standard UML sequencing operators such as sequence, concurrency, alternatives and loops.

An **actor** might be a person, a company or an organization, or a computer program or a system that interacts with a *system* (in our case *system* is a collaboration) – an actor does something and the *system* responds. An actor interacts with the system via message and data exchanges but is external to the *system* (i.e. instance of the actor is not part of the instance of its corresponding *system*). A role is a particular facet of an actor that requires specific services from the modeled *system*. For example, a person may be a Dean of a faculty and also a professor in a classroom and hence there are two roles associated with this actor – a Dean role and a Professor role. In our text, we use actors and components interchangeably. In general, each system component has some role to play, but multiple roles may be allocated to the same system component. We adopt Bochmann's [8] assumption that the allocation of the role to components is predetermined.

Informally, an **event** is something that happens and affects the system. UML, [62] defines events as "a set of possible occurrences; an occurrence is something that happens that has some consequence within the system". Events may or may not cause a state change depending on the conditions of a state. Examples of events are reception of a message, the fact that some condition becomes true, for instance, that the temperature increases above a threshold value, or the occurrence of a time-out (expiration of a timer). We consider here only the starting and ending events representing the starting and ending of execution of roles in a collaboration. These events are further discussed in Section 4.3.

4.2 Describing Collaborations with Partial Orders

It is a common practice to model distributed systems in terms of loosely coupled components modeled as communicating state machines, using languages such as Input/Output finite state machine (FSM), or an Input/Output Automata (IOA): the specification is a collection of states, and from any state, valid inputs trigger the production of an output and possibly a state change. FSM supports queues while IOA does not. State machines with variables, such as SDL [72] and UML State Machines [62], are called extended state machines. Extended state machines make use of the underlying formalism to model much more complex problems than is practical without including extended state variables. For example, let's say the behaviour of a keyboard depends on the number of keystrokes and after 5000 keystrokes, the keyboard enters a final state. Without state variables, each keystroke would be modeled as a state(for example, pressing a keystroke in state state_200 would lead to state state_201), which is clearly impractical. With state variables, there can be a key-stroke-counter variable which could be

initialized to 5000 and decrement by every keystroke until the key-stroke-counter reaches zero, and the keyboard then enters the final state.

These formalisms are adequate when specifying a sequential system. To specify behaviour of a concurrent system, an Input/Output FSM would require listing all possible ordering (combinations) of inputs and outputs, which is simply not practical. This would lead to a specification that would be quite large and difficult to create, hard to interpret and thus to understand, and whose size would make it difficult to analyze. Other models such as multi-port FSMs [51] could potentially be used, where several distributed ports are considered and at a given port, an input can generate concurrent outputs at different ports. However, this model is still intrinsically sequential regarding the inputs, which must be specified one at a time (although one such single input can generate several, concurrent outputs on different ports).

Bochmann et al. [11], introduced a new FSM methodology to specify a system with some form of concurrency: Partial Order Input Output Automata (POIOA), an extension to IOAs to deal with concurrency. In a POIOA, each transition is characterized by a partially ordered set of inputs and outputs, thus capturing the causal relationships (or enforced sequencing) between these events. This allows an up-to-exponential reduction of the number of transitions in the model, as compared with an equivalent system specified as a single I/O FSM, or even a multi-port FSM. More specifically, in a POIOA, the inputs may arrive concurrently, and transitions may execute partially, in several steps, reacting to inputs as they arrive and producing outputs as soon as they can be produced.

Adapting this methodology, we model the dynamics of inputs and outputs of activities (collaborations) as partially ordered events, where dependencies may exist between these events, shown by an arc "→" where $e_1 \rightarrow e_2$ means e_2 is directly dependent on e_1.

4.3 Collaboration Behaviour Diagram

The structural aspect of a collaboration can be defined by Collaboration Diagrams specified in UML [62] but we believe there is a need to define the semantics for the dynamic behaviour. To that end, we introduce the **Collaboration Behaviour Diagram** (CBD) to model the dynamic behaviour of UML Collaborations. We define the syntax for the Collaboration Behaviour Diagram very similar to the syntax of well-structured UML Activity Diagram defined in UML [62], but there are some differences. We model a partially ordered set of starting events and ending events for each role in the collaboration. **Starting events** are events that mark the beginning of the execution of actions by a given role in the collaboration. **Ending events** are events that mark the ending of the execution of actions by that role in the given collaboration. For a given role, due to local sequencing, there will always be a dependency from the starting event of that role to the ending event of that role. However, quite often, other dependencies may exist between starting events of a role and endings events of another role, or between multiple starting events or between multiple ending events. Hence, these dependencies form a partial order between the events. This partial order then models the dynamic behaviour of the collaboration. It is important to note that it is not necessary that all roles start their execution at the same time and end their executions at the same time, a requirement imposed for UML Activity Diagrams.

In addition, we use the concept of **initiating events** – these are those starting events, for which there is no preceding event in the partial order among the starting event. An initiating event marks the beginning of a collaboration. Note, that a given collaboration may have more than one initiating event. Similarly, **terminating events** are those ending event, for which there is no succeeding event in the partial order among all ending events. Note, that a given collaboration may have more than one terminating event. These concepts were introduced in [8].

A UML class diagram showing these concepts is given in Figure 9. We note that the distinction between starting and initiating (and between ending and terminating) events is important for modeling strict sequencing, as described in Section 5.1.

For the graphical representation of the dynamic behavior, we use a notation similar to UML Activity diagrams, illustrated in Figure 10a, where starting and ending events are shown as unfilled circle while initiating and terminating events are shown as filled-in circles.

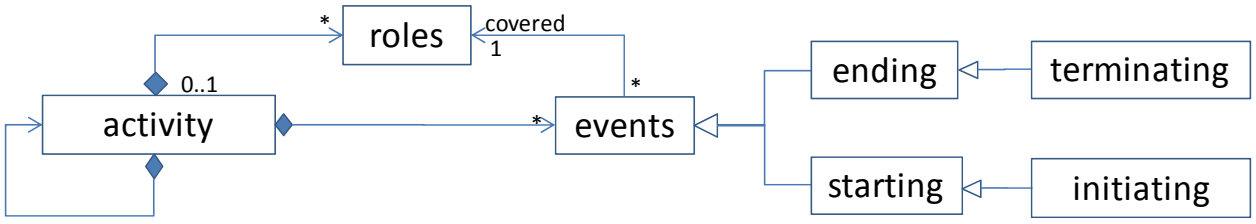


Figure 9 – Event Relationship Meta-Model

There is some similarity between our partial-order modeling and POIOA [11]. POIOA models input and output events. These events are typically external messages / signals which start and end an execution. They do not relate these events to the behaviour of any role.

However, these events can be considered as the initiating and terminating events of our modeling paradigm, CBD. In our modeling paradigm, we take it a step further, and recognize the fact that some roles may not be waiting for an external message to start their actions but are waiting for a message from another component within the same collaboration. This is the case for role R_2 in Figure 10a, as role R_2 waits for a message from role R_1 to start its execution.

In Figure 10a, events i_1 and o_1 are initiating and terminating events while i_2 and o_2 are starting and ending events, of roles R_1 and R_2, respectively. As i_1 and o_1 are events of the same role R_1, o_1 must occur after i_1 due to local sequencing (the same is true for i_2 and o_2). The starting events that are not initiating events must be triggered by a message received from a role that has an initiating event (directly / indirectly). In this example, since i_1 is the only initiating event, all events, including i_2, must occur after i_1. A *direct* dependency is shown by "→", where $i_1 \rightarrow i_2$ means i_2 is dependent on i_1. Due to the transitivity property, the relationships $i_1 \rightarrow i_2$ and $i_2 \rightarrow o_2$, leads to *indirect dependency* $i_1 \rightarrow o_2$ shown by the dashed arrow "----->."

Collaborations can be comprised of other sub-collaborations. Quite often details of sub-collaborations are provided (or known) and hence we can model the sub-collaborations and the dependencies amongst these sub-collaborations such as in Figure 10b (in this example, we assume the dependencies are a resultant of a strict sequence where execution of all the roles needs to be completed before a "dependent" collaboration may start). Other times, only the dependencies between the roles are known such as in Figure 10c and the inner details are hidden. Note: example in Figure 10c can be obtained from Figure 10b using the transitivity property.

To keep our diagrams clutter free, we do not necessarily model all the *indirect* dependencies.

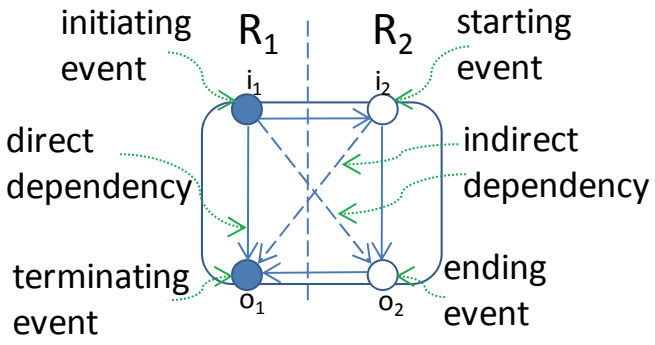


Figure 10a – Activity represented as a CBD

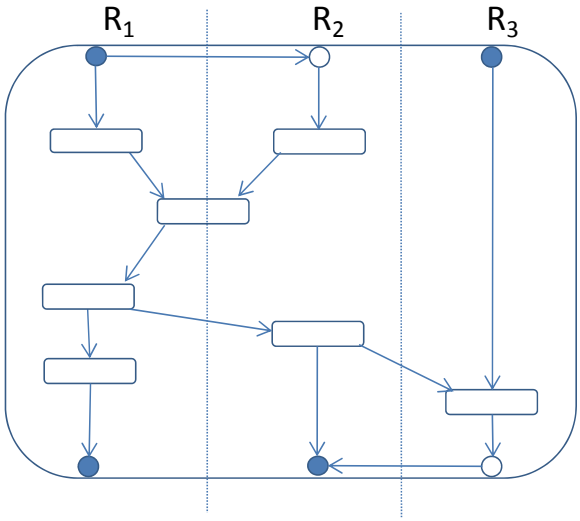


Figure 10b – Abstract Collaboration with details

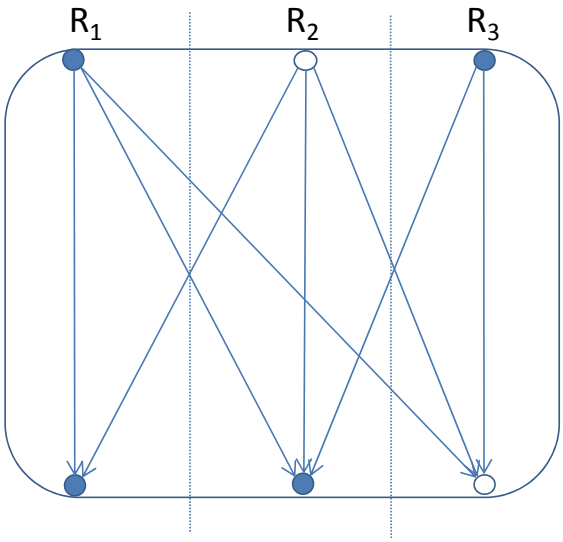


Figure 10c – Dependencies due to Transitivity

4.3.1 Modeling with CBD – an Example

Figure 11a models a fictitious Travel Management System, where a client (C) uses a Travel Management System (M) to search for flights and hotels. M delegates this task to an Air Travel Management System (A) and a Hotel Management System (H) in sub-collaboration *ProcessFlights* (PF) and sub-collaboration *ProcessHotels* (PH). In the *ProcessFlights* sub-collaboration, A inquires the flight companies, Reliable Airways (R) and Fast Airways (F), which process the request and return the results back to A, who informs M of their available options. The process is similar for the *ProcessHotels* sub-collaboration with roles E and P.

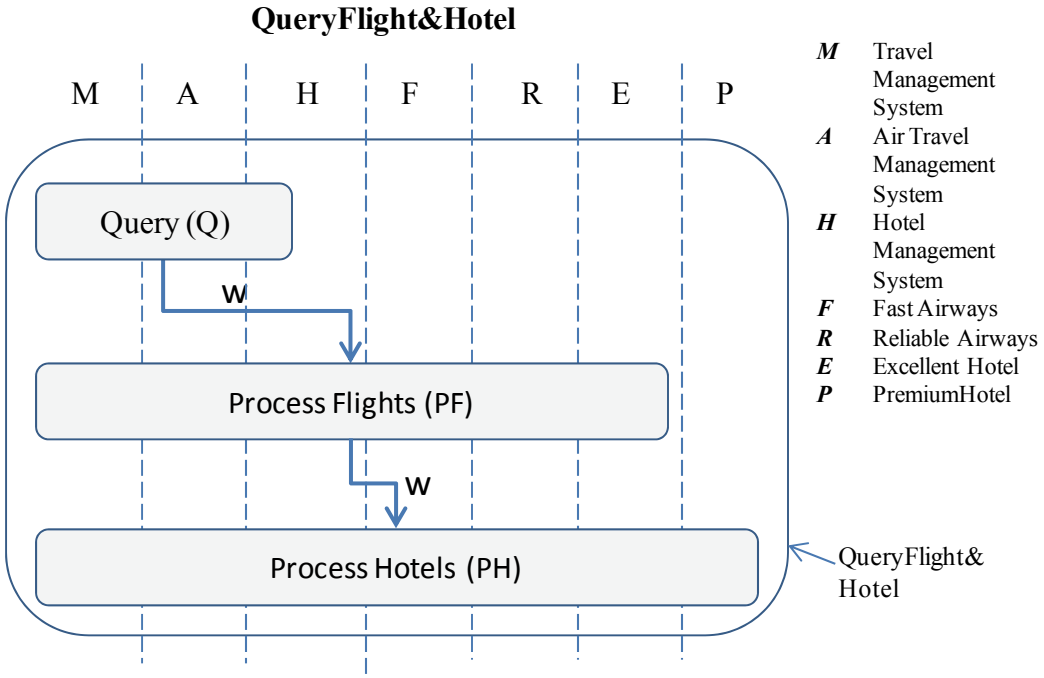


Figure 11a – Composite Collaboration *QueryFlight&Hotel*

In Figure 11b, we show all the dependencies between the various events by arcs. The initiating and terminating events are shown by filled-in circles while the other starting and ending roles are shown by un-filled circles. The three sub-collaborations, *Query*, *ProcessFlights* and *ProcessHotels* are abstracted by *QueryFlight&Hotel*. Note: the starting event of a role in the abstract collaboration is the same event as the starting event of the first collaboration that role is involved in; i.e. $_{S}i_{M} = {}_{Q}i_{M}$, $_{S}i_{A} = {}_{Q}i_{A}$, $_{S}i_{H} = {}_{Q}i_{H}$, $_{S}i_{F} = {}_{PF}i_{F}$, $_{S}i_{R} = {}_{PF}i_{R}$, $_{S}i_{E} = {}_{PH}i_{E}$, and $_{S}i_{P} = {}_{PH}i_{P}$. Similar, the ending events of a role in the abstract collaboration is the same event as the ending

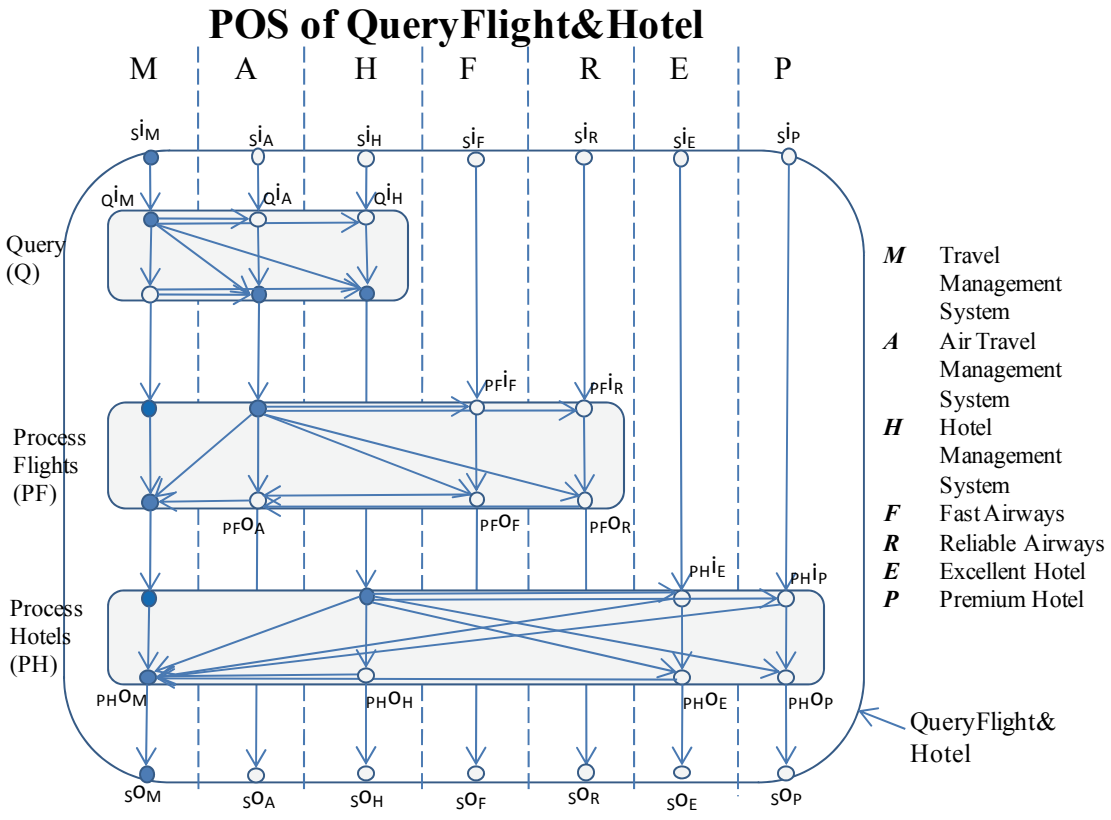


Figure 11b – CBD of *QueryFlight&Hotel* collaboration

event of the last collaboration that role is involved in, i.e., ${}_{PH}o_M = {}_{S}o_M$, ${}_{PF}o_A = {}_{S}o_A$, ${}_{PH}o_H = {}_{S}o_H$, ${}_{PF}o_F = {}_{S}o_F$, ${}_{PF}o_R = {}_{S}o_R$, ${}_{PH}o_E = {}_{S}o_E$, and ${}_{PH}o_P = {}_{S}o_P$.

Let us analyze the weak sequencing between *Query* and *ProcessFlights*. As discussed in Section 2.3.1, a weak sequence implies that *ProcessFlights* may start its execution as soon as the involved roles become available and need not to wait for execution from all the roles in *Query* to complete. Hence, even if all the actions in *Query* have not concluded, *ProcessFlights* may potentially start to execute, for example, once role A completes its execution in sub-collaboration *Query*, it can start its execution in *ProcessFlights* even if role H is still executing some action in *Query*. Similarly, role H, once having completed its execution in sub-collaboration *Query*, may start its execution in *ProcessHotels*, even if role A is still executing some actions in sub-collaboration *Query*. In contrast, had there been strict sequencing between *Query* and *ProcessHotels*, then no execution in *ProcessHotels* could have started until all the actions in *Query* had ended.

4.4 Notation

For the remainder of the thesis, we use the following notations:

- Upper case letters denote collaborations (ex. A and B would refer to collaboration A and B)
- R(X) – set of the roles involved in collaboration X
- r_x – refers to a role of collaboration X, where $r_x \in$ R(X).

- I(X) – the set of starting events of collaboration X
- O(X) – the set of ending events of collaboration X
- T_e –the time of the occurrence of the event e
- ${}_X\alpha$ – "α" defines any of the above characteristics in activity "X". *ex.* ${}_D T_e$ is the time of the occurrence of the event e in activity D

4.5 Determining Dependencies

Often there is little or no information about the performance of a sub-activity or the dependency relationships between the involved sub-collaborations and roles. To overcome this, we have devised a methodology to obtain the delays between the starting and ending events of the involved roles in a sub-activity. With this methodology, we also realize the dependencies (if any exists) between the involved roles.

Bochmann et al. in [11] introduces Partial Order Input/Output Automata (POIOA), which are automata with a finite number of states while each transition is associated with a labelled partially ordered set of input and output events as described in Section 4.2. [11] describes a method for testing a given implementation, assumed to be modeled as POIOA, to determine whether it realizes the partial-order relationships between the input and output events that are defined by a given behaviour specification, also given in the form of a POIOA. This is done by discovering unspecified output faults, missing output and unsupported input, by comparing the results produced by the implementation to the values predicted by the specification.

The testing method proceeds as follows – we model here the input (output) events of a POIOA by the starting (ending) events of the collaboration, and assume that all input events are independent of one another. In a given collaboration *A*, for each starting event i ∈ I(*A*), we perform the following two steps:

Step 1: Apply all starting events $i_m \in (I(A) \setminus \{i\})$ and observe all ending events that will occur. Call this set O1.

Step 2: Apply the starting event i and observe all ending events that will occur. Call this set O2.

Note: All ending events in O1 have no partial-order dependency on starting event i, while the ending events in O2 have a partial-order dependency on starting event i. The union of O1 and O2 must be equal to O(*A*).

Also, in Step 1, one waits for all the dependent events to occur – the maximum amount of time one should wait for all the dependent events to occur is application-specific. For example, in a networking environment, the maximum wait time for all the events, where the events are the arrival of data packets, could be 30 seconds. Possibly, this time delay could be defined in the specification of the given system.

We propose to extend this testing procedure by collecting performance measurements at the same time, as follows: For the occurrence of each ending event o ∈ O2, we measure the delay

between the time of occurrence of the starting event i and the occurrence of the ending event o. We call this delay the **Nominal Execution Time Delay** of o after i, written as Δ^{i}_{o}.

4.6 Nominal Execution Time Delay (NETD)

Definition: Nominal Execution Time Delay (NETD) of event o after event i for a collaboration A, where i ∈ I(*A*) and o ∈ O(*A*), is the time delay between event i and event o in collaboration *A*, provided that all the other starting events I'(A), where I'(A) = (I(A) \ {i}) of collaboration A have occurred a *long* time before event i (note: "\" represent "set minus"). The NETD is written as ${}_{A}\Delta^{i}_{o}$.

All the actions in A have a dependency on at least one starting event of A, i.e., all the actions in A are triggered by at least one starting event. Occurrence of starting events in I'(A) *long* time ago ensures that all the actions in A that had a dependency on I'(A) would have the time to execute completely. Hence, any remaining actions in A are dependent only on i.

It is clear that that if a dependency exists between two events i and o, where o is dependent on i, then the NETD between these two events will always be a positive real number:

$$ {}_{A}\Delta^{i}_{o} \geq 0 \text{ for all events o dependent on i} \qquad (11) $$

Note: We consider that there are cases where a dependency exists between events but their NETD is zero. For example, in local sequencing, if there is no execution between a starting event and the ending event of the same role, then this results in a dependency with a NETD with

the value zero. Another example is the NETD between an ending event of a collaboration and the starting event of the successor collaboration of the same role. As mentioned in Section 4.3.1, these two events represent the same moment in time and hence a dependency exists but their NETD is zero.

In the above test methodology, we note that the ending events o' in O1 are not dependent on the starting event i. We say that, NETD between two events o' and i, where event o' is not dependent on event i, is:

$$ {}_{A}\Delta^{i}_{o'} = -\infty \tag{12} $$

which means that o' may occur much earlier than i.

4.7 Delays for a Given Activity

In a given collaboration D, if the starting events are generated in an arbitrary order with arbitrary timing, it is clear that the time delay between the occurrence of a starting event *i* and a dependent ending event *o* can never be shorter than the NETD ${}_{D}\Delta^{i}_{o}$ measured during the above testing procedure. The time of the ending event has to be greater than or equal to the sum of time of the starting event and the NETD between that starting event and the ending event. Hence, we have the following relationship:

$$ {}_{D}T_{o} \geq {}_{D}T_{i} + {}_{D}\Delta^{i}_{o}, \tag{13} $$

Generalizing this further, we note that the ending event *o* can only occur after all the starting events on which it depends have already occurred, that is (13) must be satisfied for all the starting events. Hence, the time of the ending event *o* can be calculated as:

$$_{D}T_{o} \geq \max_{i \in I(D)} (_{D}T_{i} + _{D}\Delta^{i}_{o}), \tag{14a}$$

The time of the ending event o can be larger or equal to the sum of the time of the starting event and the delay from the starting to the ending event. The relation "larger" is due to possible sharing of resources, or other unexpected delays.

If we assume there are no shared resources (i.e., CPU sharing or such) and no queuing delay, and delays involved are only that of the execution of actions from the starting to the ending event, then the time of the output can be calculated as:

$$_{D}T_{o} = \max_{i \in I(D)} (_{D}T_{i} + _{D}\Delta^{i}_{o}), \tag{14b}$$

Due to (11), both (14a) and (14b) will always result in a positive real number.

It is important to note that the Nominal Execution Time Delay (NETD) defined above depends on a particular control flow path that was followed during the execution of a collaboration. This particular flow path could depend on many things – path chosen during an alternative, the number of times a sub-collaboration is executed within a loop etc. Therefore we write $^{(cp)}{}_{D}\Delta^{i}_{o}$ for the NETD between *i* and *o* of collaboration D for a particular control flow path *cp*.

4.8 Types of Delays

For the modeling of the performance of collaborations, we consider that the NETD that capture the performance of a collaboration could be of one three types: **fixed delays, range of delays**, and delays defined in terms of probability distributions – **stochastic delays**.

4.8.1 Stochastic Delays (SD)

In this case, we assume that the execution delays (measured or specified) are of stochastic nature and are defined by a probability distribution that could be measured by performing a large number of delay measurements. We call these **stochastic delays,** ${}_{(stoc)}{}^{(cp)}{}_{D}\Delta^{i}_{o}$. We recognize that stochastic delays could follow any kind of distribution. Hence, for our work in this thesis, we provide general formulas that can be applied to any kind of distribution. Also, as normal distributions can be used to approximate most if not all kinds of distributions under certain conditions due to the Central Limit Theorem, we provide in Chapter 6 approximations for stochastic delays in terms of normal distributions.

4.8.1.1 Special Case of Stochastic Delay - Fixed Delay (FD)

If a given NETD has a stochastic delay where the distribution is a Dirac Delta function, then we can say that the NETD has a determinist duration or a ***fixed delay (FD)*** ${}_{(fixed)}{}^{(cp)}{}_{D}\Delta^{i}_{o}$. This means that the delay for a given control flow path *cp* always results in the same value provided the starting conditions (or initial states) are the same for all participating components.

Fixed delays are commonly used when there is a need to specify performance for a single control flow path for hard real time control systems. For example, a performance requirement could be that a traffic light takes exactly 45 seconds to go from green to red, exactly another 20 seconds to go from yellow to red.

4.8.2 Range of Delays (RD)

We say that a NETD ${}^{(cp)}{}_{D}\Delta^{i}_{o,}$ has a ***range of delays (RD)*** where the delay specified for a control flow *cp* may provide different values, and it is important to specify minimum and maximum values, written as ${}_{(max)}{}^{(cp)}{}_{D}\Delta^{i}_{o}$ *and* ${}_{(min)}{}^{(cp)}{}_{D}\Delta^{i}_{o}$ respectively. In reality, the delays may be of stochastic nature, but one is normally not interested in the precise form of the probability distribution. Range of delays is used to describe the behavior of system models for hard real-time systems, using for instance the formalization of Timed Automata.

If the range is infinitesimally small, where the ${}_{(max)}{}^{(cp)}{}_{D}\Delta^{i}_{o} - {}_{(min)}{}^{(cp)}{}_{D}\Delta^{i}_{o} \approx 0$, then this results in a Fixed Delay as detailed above.

4.9 Summary

In this chapter, we discuss various elements of a distributed system, such as collaborations, actors and events. We review partially ordered sets (posets) and, based on UML Activity Diagrams and posets, we introduce Collaboration Behaviour Diagrams, a modeling paradigm to model the behaviour of a system with multiple independent starting and ending events. We also introduce a testing methodology to determine dependencies and delays between various starting

and ending events in a given collaboration and introduce the notation of NETDs, which define stochastic, fixed or a range of delays.

5. PERFORMANCE CHARACTERISTICS OF COMPOSITE COLLABORATIONS

Performance of workflows and activities has been explored by various authors. Chuang et al. discussed a stochastic Petri-net workflow model to propose various performance formulas for basic routing pattern of a workflow system [84]. Li et al. extended the Workflow net (WF-net) with timing information to provide a formal framework for modeling and performance analysis [48]. In their work, they proposed a method for computing the lower bound of the average turnaround time of transaction instances in a given workflow. Similar work has been done by Hao et al. [30], Wang et al. [84], and Lazowska et al. [47]. McNeile [56] modeled and analyzed end-to-end workflow delays but their analysis assumed that all the components are available at the beginning of the workflow, yielding a single workflow delay.

The above mentioned works assume a workflow implemented as an execution of activities implemented by a single role. This is far from reality. A typical workflow includes several roles, while each role may participate in several workflows. Complexity increases when analyzing performance of a workflow involving several roles.

With a single role participating, it is inherently assumed that the single component becomes available at the starting time of the execution of the workflow. Also, the ending event of the collaboration is only dependent on the starting event of the same role as there is no other role involved i.e. local sequencing. Dependencies between various events exist either due to

local sequencing or due to the sequencing operators. Performance is analyzed based on these dependencies.

With a workflow involving multiple roles, the very first concern is the starting time of each role involved in the given workflow. The assumption that all the roles start at the same time is unrealistic. Secondly, within a given collaboration, dependencies need to be identified which may exist between various events of different roles. Based on these dependencies, then performance can be analyzed for not only events involved in local sequencing but also between events of one role and events of another role. As such we explore well-structured collaborations with multiple roles, sequenced with standard UML operators to yield a global collaboration, describing an abstract service.

We proposed a general formula to determine the time of the ending event of a *role* relative to the time of the starting event of another (or the same) *role* and the NETD between these two events in Section 4.7. For example, in the Travel Management System from Section 4.3.1, in the collaboration "QueryFlight&Hotel", the time of the ending event for all the involved roles can be calculated based on the time of the starting events and the associated NETDs. Similarly, for the abstract collaboration "Query", the time of the ending events are again dependent on the time of the starting events and the associated NETDs. However, "QueryFlight&Hotel" collaboration is composed of "Query", "ProcessFlight" and "ProcessHotel" sub-collaborations. Assuming that the performance characteristics for each sub-collaboration are known, we show in this chapter how the performance characteristics for the global behaviour can be calculated based on the

NETDs of the constituent sub-collaborations and the control flow between the different sub-collaborations.

We consider NETDs to be fixed, in a range or stochastic, as discussed in Section 4.8. This is annotated by ${}_{(mzz)D}\Delta^{w}{}_{z}$ for the delay from the starting event w ε I(D) to the ending event z ε O(D), where *mzz* is the delay type, i.e. *mzz = fixed, max, min* or *stoc.*

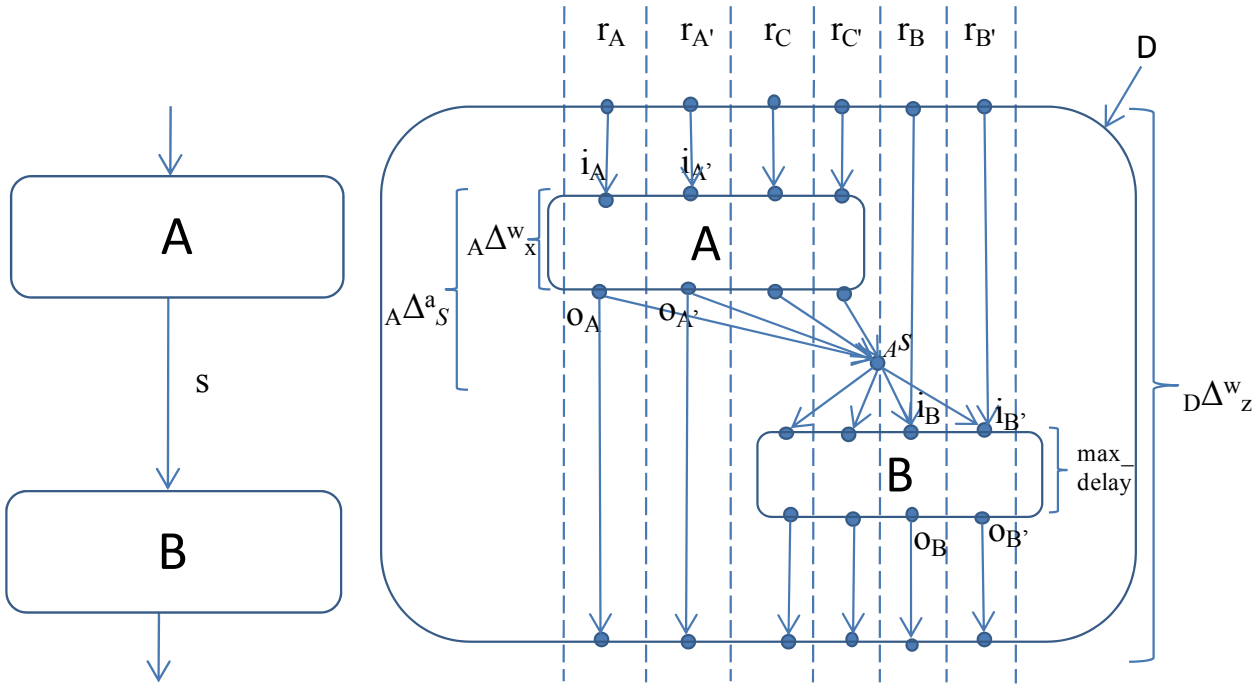
In the following, we consider various control flow operators combining sub-collaborations A and B, and forming a well-structured composite collaboration D. We assume that there are multiple roles involved in sub-collaboration A and B, where some roles may be involved only in sub-collaboration A or only in sub-collaboration B or in both. To that end, we define the following sets of roles:

- $R(D) = R(A) \cup R(B)$: the roles involved in the composite collaboration D are the roles involved in collaboration A and B.
- $R^{C}(D) = R(A) \cap R(B)$: the "common roles", that is, the roles that are involved in both collaborations A and B.
- $R^{NC}(A) = R(A) - R^{C}(D)$: the "non-common roles" of A, that is, the roles involved in sub-collaboration A but not in sub-collaboration B.

The delay ${}_{(mzz)D}\Delta^{w}{}_{z}$ of the composite collaboration D depends on the participation of the roles w and z in the sub-collaborations A and/or B. As such, for each operator, the formulas are classified according to this participation.

5.1 Strict Sequence

Figure 12a shows strict sequencing of sub-collaboration A followed by sub-collaboration B, also written as A ;s B, where ";s" represents strict sequence. [10] defines strict sequencing of A ;s B as "...the left subexpression must be terminated completely before execution of the service defined by the right subexpression may be started." This can be modeled by the partial order diagram of Figure 12b. We take this diagram as the definition of the strict sequence A ;s B. This diagram includes the event ${}_{A}s$, which we call Synchronization Event. This event occurs when all



a – Control Flow b – CBD of Strict Sequence

Figure 12 – Strict Sequence

roles involved in sub-collaboration A have completed executing their actions in sub-collaboration A.

We note that if a role is involved in sub-collaboration A and not in sub-collaboration B, then that role becomes available for its next collaboration as soon as it completes its execution in A and needs not wait for the completion of sub-collaboration B.

NETD: The Nominal Execution Time Delay ${}_{(mzz)\mathrm{D}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ from a starting event w ε I(D) to an ending event z ε O(D) for the composite collaboration D = A ;s B can be calculated using the formulas given in Table 2.

Table 2 – NETD ${}_{(mzz)\mathrm{D}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ for Strict Sequence: A ;s B

	Cases	Fixed Delays / Range of Delays	Stochastic Delay
1	w ∈ I(A) and z ∈ O(B)	$\max_{x \in O(A)}({}^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{x}})$ (15f) $+ \max_{y \in I(B)}({}^{(cp)}{}_{(mzz)\mathrm{B}}\Delta^{\mathrm{y}}{}_{\mathrm{z}})$	$(\prod_{\mathrm{x}\,\varepsilon\,\mathrm{O(A)}} {}^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{x}}(\mathrm{t}))$ (15s) $\otimes (\prod_{\mathrm{y}\,\varepsilon\,\mathrm{I(B)}} {}^{(cp)}{}_{(stoc)\mathrm{B}}\Delta^{\mathrm{y}}{}_{\mathrm{z}}(\mathrm{t}))$
2	w ∈ I(A) and z ∈ $\mathrm{O^{NC}(A)}$	${}^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ (16f)	${}^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}(\mathrm{t})$ (16s)
3	w ∈ $\mathrm{I^{NC}(B)}$ and z ∈ O(B)	${}^{(cp)}{}_{(mzz)\mathrm{B}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ (17f)	${}^{(cp)}{}_{(stoc)\mathrm{B}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}(\mathrm{t})$ (17s)
4	w ∈ $\mathrm{I^{NC}(B)}$ and z ∈ $\mathrm{O^{NC}(A)}$	$-\infty$ (18f)	$-\infty$ (18s)

In the following, we justify the formulas given in Table 2. We assume that all the dependencies shown in Figure 12b as arrows are associated with a zero delay and *mzz = max, min or fixed*.

- **Case (1) – w ∈ I(A) and z ∈ O(B) : the role of the starting event w is involved in sub-collaboration A and the role of the ending event z is involved in sub-collaboration B.** The delay from a starting event of any role w of sub-collaboration A to the synchronization event ($_{A}s$), is the maximum of the delays of all dependencies from the starting event w through all the ending events of A to event $_{A}s$.

Using (4) and (6) from Section 3.1.4.2, this can be calculated as follows, when the delay is a fixed delay or a range of delays:

Fixed or Range of Delays: $$^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{w}}{}_{s} = \max_{x \in O(A)} (^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{x}}) \quad (19f)$$

Stochastic Delay: $$^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{w}}{}_{s}(\mathrm{t}) = \prod_{\mathrm{x}\,\varepsilon\,\mathrm{O(A)}} {}^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{x}}(\mathrm{t}) \quad (19s)$$

To produce the ending event z, the dependencies from all the starting events of sub-collaboration B must be satisfied. Again from (4) and (6), this can be calculated as:

Fixed or Range of Delays: $$^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{s}}{}_{\mathrm{z}} = \max_{\mathrm{y} \in \mathrm{I(B)}} (^{(cp)}{}_{(mzz)\mathrm{B}}\Delta^{\mathrm{y}}{}_{\mathrm{z}}) \quad (20f)$$

Stochastic Delay: $$^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{s}}{}_{\mathrm{z}}(\mathrm{t}) = \prod_{\mathrm{y}\,\varepsilon\,\mathrm{I(A)}} {}^{(cp)}{}_{(stoc)\mathrm{B}}\Delta^{\mathrm{y}}{}_{\mathrm{z}}(\mathrm{t})) \quad (20s)$$

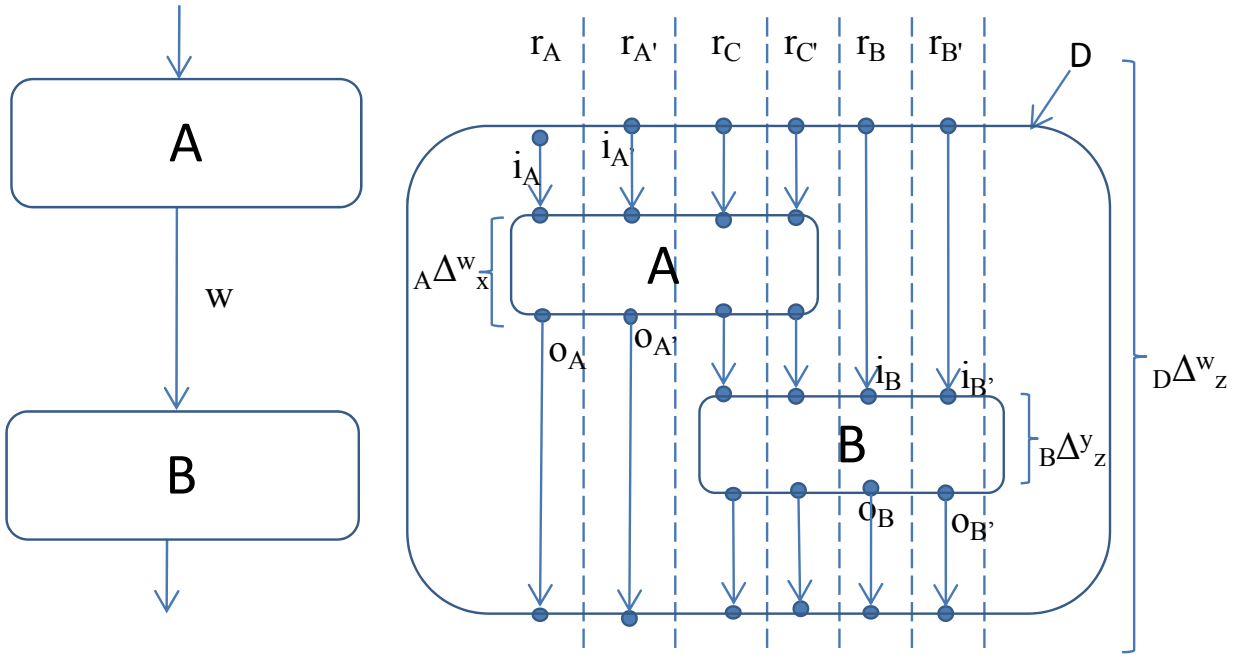
It is clear from Figure 12 that the total time delay $_{\mathrm{D}}\Delta^{w}{}_{z}$ is the resultant of $^{(cp)}{}_{\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{s}}$ and $^{(cp)}{}_{\mathrm{A}}\Delta^{\mathrm{s}}{}_{\mathrm{z}}$ in series. When the delay is either fixed or range of delays, then using (1), $_{(mzz)\mathrm{D}}\Delta^{w}{}_{z}$ is the sum of $_{(mzz)}{}^{(cp)}{}_{\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{s}}$ and $_{(mzz)}{}^{(cp)}{}_{\mathrm{B}}\Delta^{\mathrm{s}}{}_{\mathrm{z}}$. Again, similarly, if the delay is a distribution, then using (2), $_{(stoc)\mathrm{D}}\Delta^{w}{}_{z}(\mathrm{t})$ is the convolution of $_{(stoc)}{}^{(cp)}{}_{\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{s}}(\mathrm{t})$ and $_{(stoc)}{}^{(cp)}{}_{\mathrm{B}}\Delta^{\mathrm{s}}{}_{\mathrm{z}}(\mathrm{t})$.

- **Case (2) – $w \in I(A)$ and $z \in O^{NC}(A)$: role of the starting event w is involved in sub-collaboration A, and the role of the ending event z is involved in sub-collaboration A and not in sub-collaboration B.** As sub-collaboration B is not involved, the NETD from w to z of collaboration D is the NETD from w to z of sub-collaboration A as all the other starting events have occurred a long time ago, leaving only the dependency from w to z in sub-collaboration A.
- **Case (3) – $w \in I^{NC}(B)$ and $z \in O(B)$: the role of the starting event w is involved in sub-collaboration B and not in sub-collaboration A and the role of the ending event z is involved in sub-collaboration B.** Similar to case (2), all the other starting events have occurred a long time ago, leaving only the delay from w to z in sub-collaboration B, and hence the NETD of collaboration D is that of the NETD of sub-collaboration B.
- **Case (4) – $w \varepsilon I^{NC}(B)$ and $z \varepsilon O^{NC}(A)$: the role of the starting event is involved in sub-collaboration B and not in sub-collaboration A, and the role of the ending event is involved only in sub-collaboration A and not in sub-collaboration B.** Sub-collaboration A is followed by sub-collaboration B, and not the other way around. As there is no causal dependency from the event w of sub-collaboration B to the event z of sub-collaboration A, there is no delay from events of sub-collaboration B to events of sub-collaboration A, which is encoded by the $-\infty$ value (see Section 4.6).

5.2 Weak Sequence

In contrast to strict sequencing, Figure 13 shows weak sequencing of sub-collaboration A followed by sub-collaboration B, also written as A ;w B where ";w" represents weak sequence.

[8] defines weak sequencing A1 ;w A2 as "...each system component locally applies sequencing to the local sub-collaborations of A1 and A2, that is, a component may start with sub-collaborations that belong to A2 as soon as it has completed all its local sub-collaborations that are part of A1."



a – Control Flow b – CBD of Weak Sequence

Figure 13 – Weak Sequence

A partial order diagram of A ;w B is shown in Figure 13b, where components in sub-collaboration B start their processing as soon as they have completed their processing in sub-collaboration A. We takes this diagram as the definition of the weak sequence A ;w B. For the roles not involved in sub-collaboration A, and only involved in sub-collaboration B, they may

start their execution in sub-collaboration B without any delay stemming from sub-collaboration A, possibly even before sub-collaboration A begins its execution.

NETD: The Nominal Execution Time Delays for the composite collaboration D = A ;w B from the starting event w ∈ I(D) to the ending event z ∈ O(D), ${}_{(mzz)\mathrm{D}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$, are given by the formulas in Table 3:

Table 3 – NETD ${}_{(mzz)\mathrm{D}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ for Weak Sequence: A ;w B

	Cases	Fixed Delays / Range of Delays	Stochastic Delay
1	w ∈ I(A) and z ∈ O(B)	$\max_{x\varepsilon Ic(D)} ({}^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{x}} + {}^{(cp)}{}_{(mzz)\mathrm{B}}\Delta^{\mathrm{x}}{}_{\mathrm{z}})$ (21f)	$\prod_{x\varepsilon Ic(D)} ({}^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{x}}(t) \otimes {}^{(cp)}{}_{(stoc)\mathrm{B}}\Delta^{\mathrm{x}}{}_{\mathrm{z}}(t))$ (21s)
2	w ∈ I(A) and z ∈ O^{NC}(A)	${}^{(cp)}{}_{(mzz)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ (22f)	${}^{(cp)}{}_{(stoc)\mathrm{A}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}(t)$ (22s)
3	w ∈ I^{NC}(B) and z ∈ O(B)	${}^{(cp)}{}_{(mzz)\mathrm{B}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$ (23f)	${}^{(cp)}{}_{(stoc)\mathrm{B}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}(t)$ (23s)
4	w ∈ I^{NC}(B) and z ∈ O^{NC}(A)	$-\infty$ (24f)	$-\infty$ (24s)

Similar to strict sequencing, in the following, we justify the formulas given in Table 3. We again assume that all the dependencies shown in Figure 13 by an arrow are associated with a zero delay and *mzz = max, min or fixed.*

Note: The justification is the same for stochastic delays that is of the fixed and range of delays, and hence we justify the formulas for only the fixed and range of delays and not for the stochastic delays.

- **Case (1) – w ∈ I(A) and z ∈ O(B) : role of the starting event w is involved in sub-collaboration A and the role of the ending event z is involved in sub-collaboration B.** There must exist some common role(s) between A and B for a dependency (and hence a delay) to exist from the starting event in A to the ending event in B. As the ending event in sub-collaboration A is the same event as the starting event in sub-collaboration B for each common role x, the time of the ending event and the starting event is the same. We can then conclude from (2) that the delay for the "dependency path" through role x, is the sum of two delays:

$$_{(mzz)}path\_delay\_through\_x = {}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{x} + {}^{(cp)}{}_{(mzz)B}\Delta^{x}{}_{z} \qquad (25f)$$

To find the delay to ending event z, traversing each path from w → x and x → z will ensure all the dependencies that z depends on will be satisfied, resulting in:

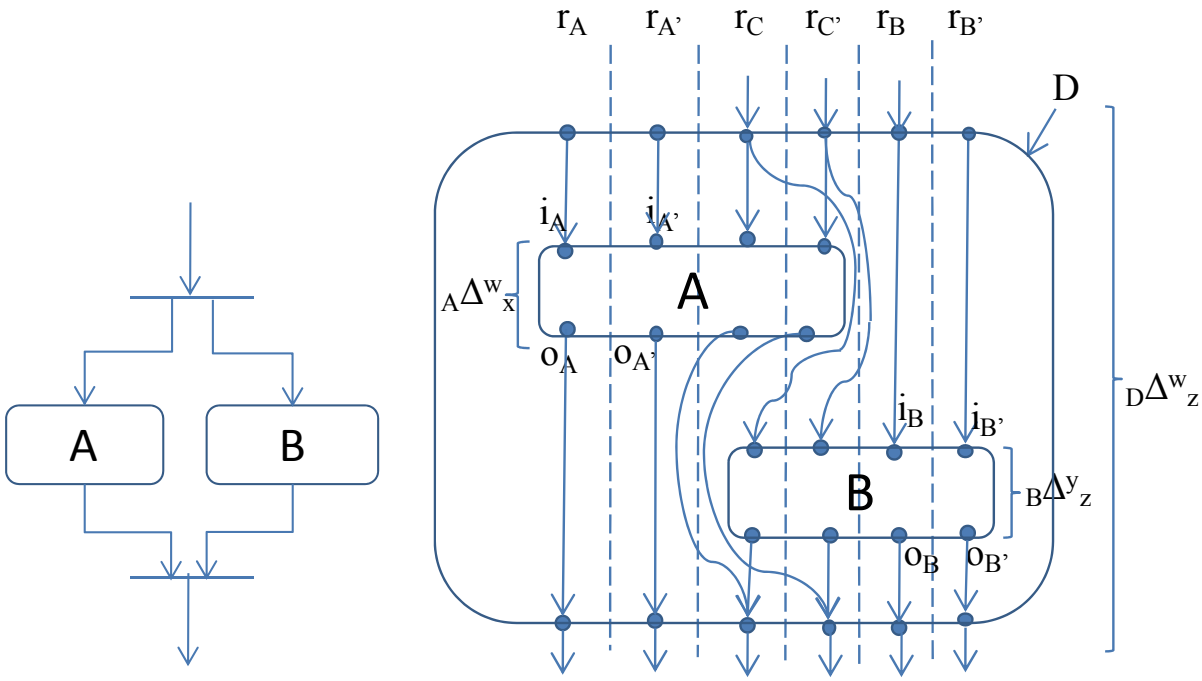
$${}^{(cp)}{}_{(mzz)D}\Delta^{w}{}_{z} = \max_{x \,\varepsilon\, Ic(D)} ({}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{x} + {}^{(cp)}{}_{(mzz)B}\Delta^{x}{}_{z}) \qquad (26f)$$

- **Case (2) – w ∈ I(A) and z ∈ O^{NC}(A): role of the starting event w is involved in sub-collaboration A, and the role of the ending event z is involved in sub-collaboration A and not in sub-collaboration B.** This case is similar to that of Case (2) of strict sequencing.
- **Case (3) – w ∈ I^{NC}(B) and z ∈ O(B): the role of the starting event w is involved in sub-collaboration B and not in sub-collaboration A and the role of the ending event z is involved in sub-collaboration B.** This case is similar to that of Case (3) of strict sequencing.

- **Case (4) – $w \in I^{NC}(B)$ and $z \in O^{NC}(A)$: the role of the starting event is involved in sub-collaboration B and not in sub-collaboration A, and the role of the ending event is involved only in sub-collaboration A and not in sub-collaboration B.** This case is similar to that of Case (4) of strict sequencing.

5.3 Concurrency

Figure 14 shows concurrent execution of sub-collaborations A and B, written as A || B. The non-common roles become available for their next collaboration, as soon as their execution has completed in their respective sub-collaborations.



a – Control Flow b – CBD of Concurrent Sub-Collaborations

Figure 14 – Concurrency

We assume there are infinite processes each with their own processor for any given common role. Hence, one implementation of a common role does not have any effect on the performance of a second implementation of the same common role. However, the common roles are available for their next collaboration only when execution of the common role has completed in both sub-collaborations, such as shown in Figure 14b.

NETD: The NETD from the starting event w to the ending event z, ${}_{(mzz)D}\Delta^{w}{}_{z}$, for collaboration D = A || B can be calculated using the formulas in Table 4.

Table 4 – NETD ${}_{(mzz)D}\Delta^{w}{}_{z}$ for Concurrency: A || B

	Cases	**Fixed Delays / Range of Delays**	**Stochastic Delays**
1	$w \varepsilon I^{C}(D)$ and $z \varepsilon O^{C}(D)$	$\max({}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{z}, {}^{(cp)}{}_{(mzz)B}\Delta^{w}{}_{z})$ (27f)	${}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{z}(t) * {}^{(cp)}{}_{(stoc)B}\Delta^{w}{}_{z}(t)$ (27s)
2	$(w \varepsilon I^{NC}(A)$ and $z \varepsilon O(A))$ or $(w \varepsilon I(A)$ and $z \varepsilon O^{NC}(A)$	${}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{z}$ (28f)	${}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{z}(t)$ (38s)
3	$(w \varepsilon I^{NC}(B)$ and $z \varepsilon O(B))$ or $(w \varepsilon I(B)$ and $z \varepsilon O^{NC}(B))$	${}^{(cp)}{}_{(mzz)B}\Delta^{w}{}_{z}$ (29f)	${}^{(cp)}{}_{(stoc)B}\Delta^{w}{}_{z}(t)$ (29s)
4	$(w \varepsilon I^{NC}(A)$ and $z \varepsilon O^{NC}(B))$ or $(w \varepsilon I^{NC}(B)$ and $z \varepsilon O^{NC}(A))$	$-\infty$ (30f)	$-\infty$ (30f)

where *mzz = max, min or fixed*

We again assume that all the dependencies shown in Figure 14b by an arrow are associated with a zero delay. For these formulas, we consider each of the equations:

- **Case (1) – ($w \varepsilon I^C(D)$ and $z \varepsilon O^C(D)$): roles of both the starting event and the ending, are "common" roles i.e. they are involved in both sub-collaborations A and B.** For the execution of two concurrent sub-collaborations, a common role must complete its execution in both sub-collaborations before any subsequent executions can be performed by that role, as shown in Figure 14b. Since the given common role is the same for both sub-collaborations, the starting events must occur at the same time. Hence from (4) and (6), the delay is the maximum of two delays represented by the maximum of two fixed values or the product of the two distributions.
- **Case (2) – ($w \varepsilon I^{NC}(A)$ and $z \varepsilon O(A)$) or ($w \varepsilon I(A)$ and $z \varepsilon O^{NC}(A)$): either role of the starting event is a non-common role of sub-collaboration A while role of the ending event is any role of sub-collaboration A, or vice-versa.** As can be seen in Figure 14b, except for the delay from the starting event to the ending event of the common roles, the NETD between the starting event w and ending event z where ($w \varepsilon I(A)$ and $z \varepsilon O^c(A)$) or ($w \varepsilon I^c(A)$ and $z \varepsilon O(A)$) or ($w \varepsilon I^{NC}(A)$ and $z \varepsilon O^{NC}(A)$) is due only to the delay from the starting event w to the dependent ending event z. There is no other dependency which needs to be satisfied for event z to occur. Hence, the NETD of collaboration D is the NETD of collaboration A from w to z, that is ${}_A\Delta^w{}_z$.
- **Case (3) – ($w \varepsilon I^{NC}(B)$ and $z \varepsilon O(B)$) or ($w \varepsilon I(B)$ and $z \varepsilon O^{NC}(B)$): same as case (6) except for sub-collaboration B instead of sub-collaboration A.** This scenario is similar to the Case (6) above.

Case (4) – ($w \varepsilon I^{NC}(A)$ and $z \varepsilon O^{NC}(B)$) or ($w \varepsilon I^{NC}(B)$ and $z \varepsilon O^{NC}(A)$): either role of the starting event is a non-common role in sub-collaboration A while role of the ending event is a non-common role in sub-collaboration B, or vice-versa. As can be seen from Figure 14b,

there is no dependency from a starting event of a non-common role of collaboration A to an ending event of a non-common role of collaboration B (or A). As defined by (3), the NETD for non-dependent events is $-\infty$.

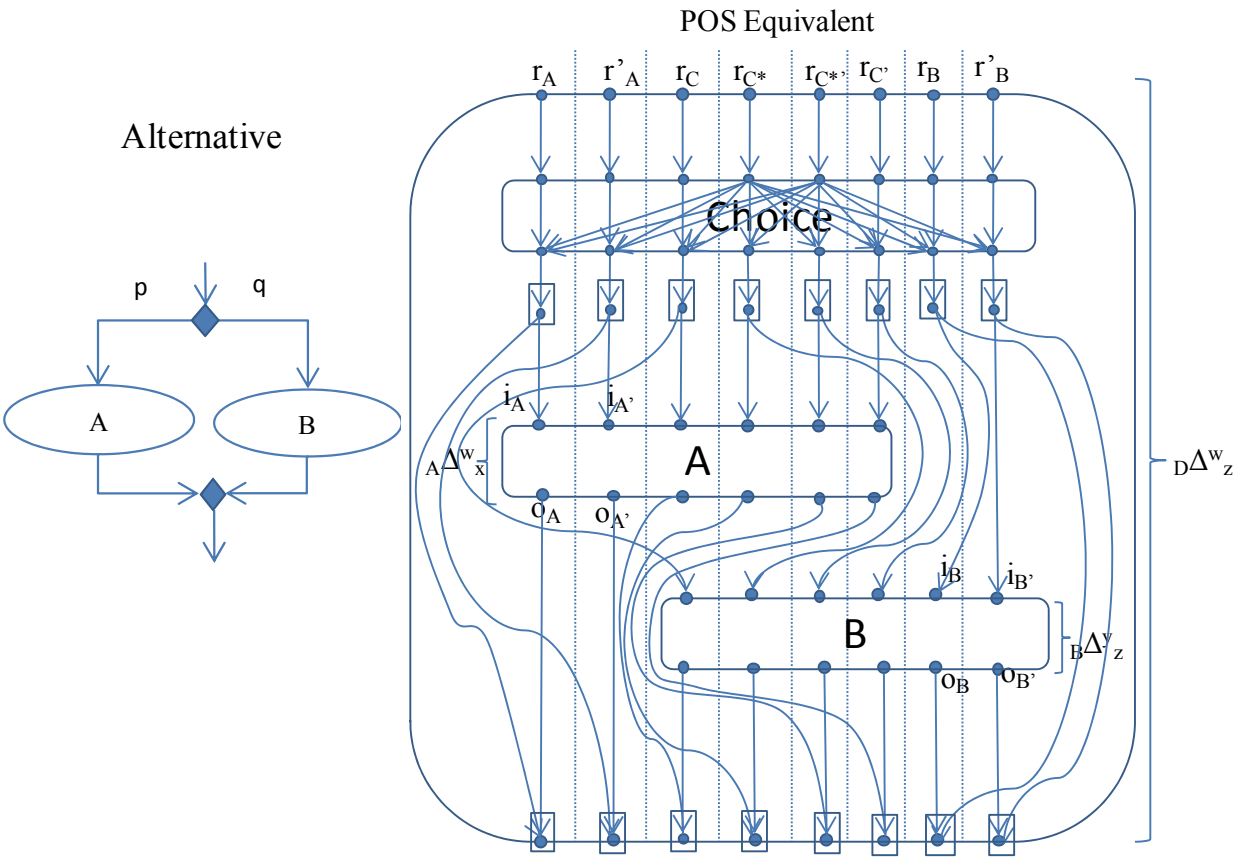
5.4 Alternatives

Figure 15 shows the alternative choice between two sub-collaborations, A and B, written as A [] B. This behaviour is abstracted as collaboration D. So far, only a single control flow path had existed for strict/weak sequence and concurrency. We had considered fixed delays, range of delays and delay distributions for a single control flow path.

A **control flow path** among collaborations defines in which order the non-refined collaborations are executed. A behaviour may have alternatives, interruptions and loops which may allow for different paths for a given behavior definition. It is clear there may exist several different paths in a collaboration – for instance, each branch of an alternative yields a different path. In general, the number of different paths is infinite. For instance, the number of times a while loop is executed may be unbounded, and the body of a loop or alternative may include embedded loops or alternatives, which results in a recursive structure of possible execution paths.

For an alternative, there can be multiple control flow paths, one flow path for each alternative path of execution.

Before either of the two collaborations execute, a choice needs to be made between the two sub-collaborations. We assume the choice is made by a set of roles C*, where C* ∈ R(D), and we assume that all the roles involved in making the choice make the same choice. No action in either sub-collaboration A or B may execute until this choice is made. We model this decision making process by adding an abstract sub-collaboration "Choice" preceding the sub-collaborations A and B, where we introduce a dependency from the starting events of C* to



a – Control Flow b – Partial Order Equivalent

Figure 15 – Alternative Operator

the ending events of all the roles in sub-collaboration Choice. This is to make sure none of the roles start their execution in either of the two sub-collaborations. Hence, sub-collaboration Choice weakly precedes with collaboration A and B. The set of roles in sub-collaboration Choice is defined by R(Choice) = C* ∪ R(A) ∪ R(B). This may introduce additional dependencies between the starting event of C* and z ∈ O(D). We assume the choice and the propagation of this choice to all the other involved roles (shown by the dependency arcs in Choice sub-collaboration) is done instantaneously, and does not add any delay. Also note that for non-initiating roles may not need explicit choice propagation.

First we analyze the sub-collaborations for a particular single control flow path and calculate fixed delays, range of delays and delay distributions and based on these delays, then we consider all possible control flow paths and calculate the range of delays and stochastic delays.

5.4.1 Consideration of a Single Control Flow Path

NETD: If we consider the control flow path for which sub-collaboration A executes, then the NETD, ${}^{(cp)}{}_{(mzz)\mathrm{D}}\Delta^{\mathrm{w}}{}_{\mathrm{z}}$, for the composite collaboration D consisting of an alternative execution such as shown in Figure 15b is obtained as shown in Table 5a. The definition for delays for the case that sub-collaboration B executes is similar to that of Table 5a and can be obtained by symmetry.

Table 5a – NETD for A [] B for a Single Control Flow Path when A executes

	Case	***Fixed and Range of Delays***	***Stochastic Delays***
1	$w \in I(C^*)$ and $z \in O(A)$	$\max_{y \in I(A)} ({}^{(cp)}{}_{(mzz)A}\Delta^{y}{}_{z})$ (31f)	$\prod_{y \varepsilon I(A)} ({}^{(cp)}{}_{(stoc)A}\Delta^{y}{}_{z}(t))$ (31s)
2	$w \in I(C^*)$ and $z \in O^{NC}(B)$	0 (32f)	$\delta(0)$ (32s)
3	$w \notin I(C^*)$ and ($w \in I(A)$ and $z \in O(A)$)	${}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{z}$ (33f)	${}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{z}(t)$ (33s)
4	$w \notin I(C^*)$ and (**w** $\in I^{NC}(B)$ **or** $z \in O^{NC}(B)$ where $w \neq z$)	$-\infty$ (34f)	$-\infty$ (34s)
5	$w \notin I(C^*)$ and (**w** $\in I^{NC}(B)$, **or** $z \in O^{NC}(B)$where $w = z$)	0 (35f)	$\delta(0)$ (34s)

where *mzz = fixed, max or min.*

We again assume that all the dependencies shown in Figure 15b by an arrow are associated with a zero delay. Also, note that collaboration A and B both are preceded by a Choice collaboration. In a Choice collaboration, there is a delay of at least zero when starting event of role w $\in C^*$ or when starting event of role w is the same as the ending event of role z i.e. w = z. In all the other cases, if there is no dependency, then the delays are $-\infty$. We consider the following equations:

Case 1 – w ∈ I(C*) and z ∈ O(A):

The role of the starting event w is involved in making the choice and the role of the ending event z is involved in A

As A is selected to execute, all the starting events of collaboration A need to wait for the choice to be made by w, causing a dependency from w to all the starting events of collaboration A, which in turn causes dependencies from all starting events of collaboration A to the ending event z. Performance of this dependency can be calculated by taking the maximum over all the roles of the starting events of A to the ending event z. This is a reduced form of equation (21).

Case 2 – w ∈ I(C*) and z ε O^{NC}(B):

The role of the starting event w is involved in making the choice and the role of the ending event z is involved only in B and not in A

As the role of the ending event is involved only in B and not in A, and considering A is selected to execute, there is no execution which will cause a delay to B, other than the choice propagation in the Choice sub-collaboration as shown in Figure 15b. We had assumed this delay to be zero and hence the delay from w to z is zero.

Case 3 – w ε I(A) and z ε O(A) and w ∉ I(C*):

The roles of the starting and the ending events, w and z respectively, are both involved in collaboration A and w is not involved in making the choice.

Since A executes, and the role of w is not involved in making the choice, all the remaining starting events have already occurred except w as per the definition of NETD in section 4.6.

Hence the only delay is from w to z. Note: if the role w is the same as role z, then the delay must be at least zero due to (11).

Case 4 – $w \varepsilon I^{NC}(B)$ or $z \varepsilon O^{NC}(B)$ where $w \neq z$

Either the role of the starting event or the role of the ending event is involved in only collaboration B and not in A, and the role of the starting event is not the same role as the role of the ending event.

If the role of the starting event is involved only in sub-collaboration B, then this role does not execute any of its actions as sub-collaboration A has been selected to execute. Hence there is no dependency to any of the roles except its own role and hence due to (11), delay is $-\infty$.

Case 5 – $w \varepsilon I^{NC}(B)$ or $z \varepsilon O^{NC}(B)$ where $w = z$

Same as Case 4, except that the role of w is the same as the role of z.

Since w is the same role as z, the delay must be at least zero due to (11). And since collaboration A executes and not B, there is no additional delay.

5.4.1.1 Consideration of a Single Control Flow Path – Reorganized

Table 5a defined delays of a single control flow path if A is executed in A [] B. Since we need to consider both cases, that A executes and that B executes, when we deal with ranges of delays or stochastic delays, we present in the following Table 5b both cases together. This leads to a much larger number of cases. Since the content of Table 5b can be obtained from Table 5a in a straightforward manner, we do not present any proofs. This table is used to aid the analysis of delays when considering multiple control flow paths.

Table 5b – NETD for A [] B for a Single Control Flow Path when A or B executes

	Case	*A Executes*	*B Executes*
1	$w \in I(C^*)$ and $z \in O^{NC}(A)$	$\max_{y \in I(A)} ({}^{(cp)}_{(mzz)A}\Delta^{y}_{z})$ (36a)	0 (36b)
2	$w \in I(C^*)$ and $z \in O^{NC}(B)$	0 (37a)	$\max_{y \in I(B)} ({}^{(cp)}_{(mzz)B}\Delta^{y}_{z})$ (37b)
3	$w \in I(C^*)$ and $z \in O^{C}(D)$	$\max_{y \in I(A)} ({}^{(cp)}_{(mzz)A}\Delta^{y}_{z})$ (38a)	$\max_{y \in I(B)} ({}^{(cp)}_{(mzz)B}\Delta^{y}_{z})$ (38b)
4	$w \notin I(C^*)$ and ($w \in I^{NC}(A)$ and $z \in O^{NC}(A)$, where $(w \neq z)$) or ($w \in I^{NC}(A)$ and $z \in O^{C}(D)$) or ($w \in I^{C}(D)$ and $z \in O^{NC}(A)$)	${}^{(cp)}_{(mzz)A}\Delta^{w}_{z}$ (39a)	$-\infty$ (39b)
5	$w \notin I(C^*)$ and ($w \in I^{NC}(A)$ and $z \in O^{NC}(A)$, where $w = z$)	${}^{(cp)}_{(mzz)A}\Delta^{w}_{z}$ (40a)	0 (40a)
6	$w \notin I(C^*)$ and (($w \in I^{NC}(A)$ and $z \in O^{NC}(B)$) or ($w \in I^{NC}(B)$ and $z \in O^{NC}(A)$))	$-\infty$ (41a)	$-\infty$ (41b)
7	$w \notin I(C^*)$ and ($w \in I^{C}(D)$ and $z \in O^{C}(D)$)	${}^{(cp)}_{(mzz)A}\Delta^{w}_{z}$ (42a)	${}^{(cp)}_{(mzz)B}\Delta^{w}_{z}$ (42b)

8	$w \notin I(C^*)$ and (($w \in I^{NC}(B)$ and $z \in O^{NC}(B)$, where $w \neq z$) or ($w \in I^{C}(D)$ and $z \in O^{NC}(B)$) or ($w \in I^{NC}(B)$ and $z \in O^{C}(D)$)	$-\infty$ (43a)	${}^{(cp)}{}_{(mzz)B}\Delta^{w}{}_{z}$ (43b)
9	$w \notin I(C^*)$ and ($w \in I^{NC}(B)$ and $z \in O^{NC}(B)$, where $w = z$)	0 (44a)	${}^{(cp)}{}_{(mzz)B}\Delta^{w}{}_{z}$ (44b)

where mzz = fixed

5.4.2 Considering Multiple Control Flow Paths – Range of Delays

Based on the formulas in Table 5b, we calculate the range of delays in Table 6 for all the possible control flow paths for the alternative operator in the composite collaboration D.

Note 1: The Choice sub-collaboration creates a dependency between all the starting events $w \in I(C^*)$, and all the ending events $z \in O(D)$, with a delay of 0. Execution of collaborations A or B may add additional delay from w to z but due to the Choice sub-collaboration, the delay is at least 0.

Table 6 – NETD for Alternative Operator for Multiple Control Flow Paths – Range of Delays

	Case	***Maximum Delay***	***Minimum Delay***
1	$w \in I(C^*)$ and $z \in O^{NC}(A)$	$\max_{y \in I(A)} ({}^{(cp)}{}_{(max)A}\Delta^{y}{}_{z})$ (45a)	0 (45b)

2	$w \in I(C^*)$ and $z \in O^{NC}(B)$	$\max_{y \in I(B)} ({}^{(cp)}_{(max)B}\Delta^{y}_{z})$ (46a)	0 (46b)
3	$w \in I(C^*)$ and $z \in O^{C}(D)$	max ((47a) $\max_{y \in I(A)} ({}^{(cp)}_{(max)A}\Delta^{y}_{z})$, $\max_{y \in I(B)} ({}^{(cp)}_{(max)B}\Delta^{y}_{z})$)	min ((47b) $\max_{y \in I(A)} ({}^{(cp)}_{(min)A}\Delta^{y}_{z})$, $\max_{y \in I(B)} ({}^{(cp)}_{(min)B}\Delta^{y}_{z})$)
4	$w \notin I(C^*)$ and ($w \in I^{NC}(A)$ and $z \in O^{NC}(A)$, where $(w \neq z)$) or ($w \in I^{NC}(A)$ and $z \in O^{C}(D)$) or ($w \in I^{C}(D)$ and $z \in O^{NC}(A)$)	${}^{(cp)}_{(max)A}\Delta^{w}_{z}$ (48a)	$-\infty$ (48b)
5	$w \notin I(C^*)$ and ($w \in I^{NC}(A)$ and $z \in O^{NC}(A)$, where $w = z$)	${}^{(cp)}_{(max)A}\Delta^{w}_{z}$ (49a)	0 (49b)
6	$w \notin I(C^*)$ and (($w \in I^{NC}(A)$ and $z \in O^{NC}(B)$) or ($w \in I^{NC}(B)$ and $z \in O^{NC}(A)$))	$-\infty$ (50a)	$-\infty$ (50b)
7	$w \notin I(C^*)$ and ($w \in I^{C}(D)$ and $z \in O^{C}(D)$)	max(${}^{(cp)}_{(max)A}\Delta^{w}_{z}$, (51a) ${}^{(cp)}_{(max)B}\Delta^{w}_{z}$)	min(${}^{(cp)}_{(min)A}\Delta^{w}_{z}$, (51b) ${}^{(cp)}_{(min)B}\Delta^{w}_{z}$)
8	$w \notin I(C^*)$ and (($w \in I^{NC}(B)$ and $z \in O^{NC}(B)$, where $w \neq z$) or ($w \in I^{C}(D)$ and $z \in O^{NC}(B)$) or ($w \in I^{NC}(B)$ and $z \in O^{C}(D)$))	${}^{(cp)}_{(max)B}\Delta^{w}_{z}$ (52a)	$-\infty$ (52b)

9	$w \notin I(C^*)$ and **(**$w \in I^{NC}(B)$ and $z \in O^{NC}(B)$, where $w = z$)	${}^{(cp)}{}_{(max)B}\Delta^{w}{}_{z}$ (53a)	0 (53b)

Formulas in Table 6 are very intuitively derived from Table 5b. The cases are the same as for the Table 5b. The maximum (/minimum) delays are the maximum (/minimum) of the two cases where A or B executes.

5.4.3 Consideration of Multiple Control Flow Paths – Stochastic Delay

We assume there is a probability p for sub-collaboration A to execute, and probability q=1-p for sub-collaboration B to execute. We can use the formulas in Table 5b to calculate the delay distribution for all possible control flow paths. To calculate the stochastic delays, we assume the NETDS of sub-collaboration A and B are also stochastic. If these delays are range of delays, we have no stochastic information. If they are fixed delays, we consider the delays as a Dirac delta distribution. The stochastic NETD for a composite collaboration D consisting of an alternate execution of sub-collaborations A and B, ${}^{(cp)}{}_{(stoc)D}\Delta^{w}{}_{z}$, is given in Table 7.

Table 7 – NETD for Alternative Operator for Multiple Control Flow Paths – Stochastic Delays

	Case	***Stochastic Delays***
1	$w \in I(C^*)$ and $z \in O^{NC}(A)$	$p * \prod_{y \varepsilon I(A)} ({}^{(cp)}{}_{(stoc)A}\Delta^{y}{}_{z}(t)) + q * \delta(0)$ (54)
2	$w \in I(C^*)$ and $z \in O^{NC}(B)$	$p * \delta(0) + q * \prod_{y \varepsilon I(B)} ({}^{(cp)}{}_{(stoc)B}\Delta^{y}{}_{z}(t))$ (55)
3	$w \in I(C^*)$ and $z \in O^{C}(D)$	$p * \prod_{y \varepsilon I(A)} ({}^{(cp)}{}_{(stoc)A}\Delta^{y}{}_{z}(t)) +$ (56) $q * \prod_{y \varepsilon I(B)} ({}^{(cp)}{}_{(stoc)B}\Delta^{y}{}_{z}(t))$

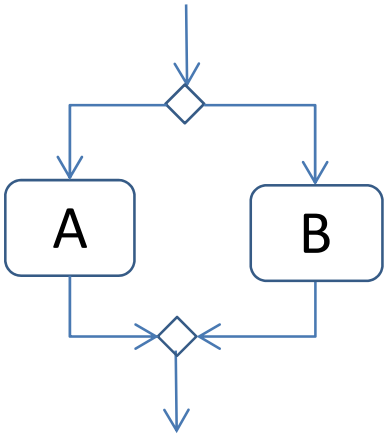
4	$w \notin I(C^*)$ and ($w \in I^{NC}(A)$ and $z \in O^{NC}(A)$, where $(w \neq z)$) or ($w \in I^{NC}(A)$ and $z \in O^{C}(D)$) or ($w \in I^{C}(D)$ and $z \in O^{NC}(A)$)	$p * {}^{(cp)}_{(stoc)}{}_{A}\Delta^{w}{}_{z}(t)$, (57a) with probability q, there is no dependency from w to z (in case B is executed)
5	$w \notin I(C^*)$ and ($w \in I^{NC}(A)$ and $z \in O^{NC}(A)$, where $w = z$)	$p * {}^{(cp)}_{(stoc)}{}_{A}\Delta^{w}{}_{z}(t) + q * \delta(0)$ (57b)
6	$w \notin I(C^*)$ and (($w \in I^{NC}(A)$ and $z \in O^{NC}(B)$) or ($w \in I^{NC}(B)$ and $z \in O^{NC}(A)$))	$-\infty$ (58)
7	$w \notin I(C^*)$ and ($w \in I^{C}(D)$ and $z \in O^{C}(D)$)	$p * {}^{(cp)}_{(stoc)A}\Delta^{w}{}_{z}(t) + q * {}^{(cp)}_{(stoc)B}\Delta^{w}{}_{z}(t)$ (59)
8	$w \notin I(C^*)$ and (($w \in I^{NC}(B)$ and $z \in O^{NC}(B)$, where $w \neq z$) or ($w \in I^{C}(D)$ and $z \in O^{NC}(B)$) or ($w \in I^{NC}(B)$ and $z \in O^{C}(D)$)	$q * {}^{(cp)}_{(stoc)}{}_{B}\Delta^{w}{}_{z}(t)$, (60a) with probability p, there is no dependency from w to z (in case A is executed)

9	$w \notin I(C^*)$ and $(w \in I^{NC}(B)$ and $z \in O^{NC}(B)$, where $w = z)$	$p * \delta(0) + q * {}_{(stoc)}{}^{(cp)}{}_{B}\Delta^{w}{}_{z}(t)$	(60b)

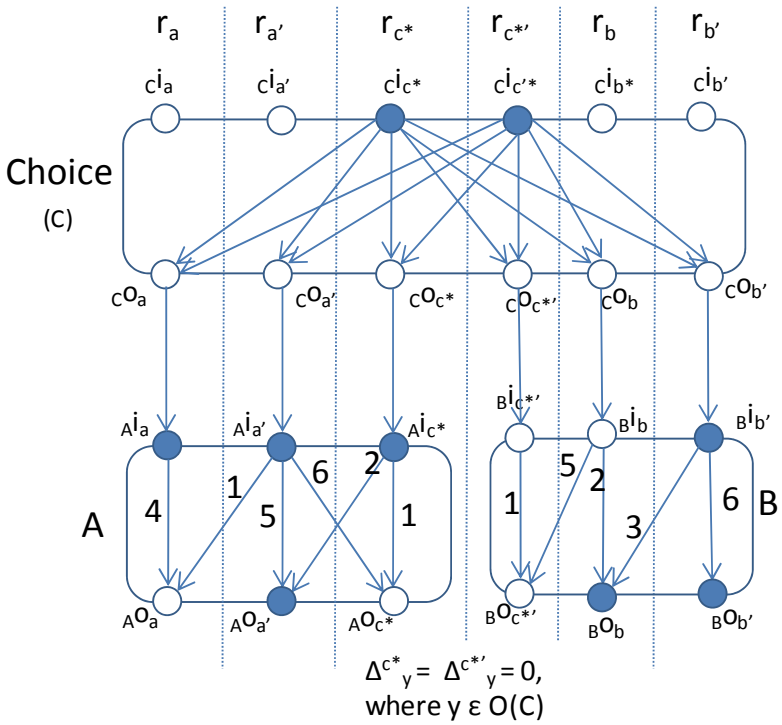
Formulas of Table 7 are the stochastic operators of the formulas defined from Table 5b. According to Equation (7), the stochastic delay is the sum of the product of the respective probability with the delay of the respective collaboration.

5.4.4 Example of an Alternative Operator when Considering a Single Control Flow Path

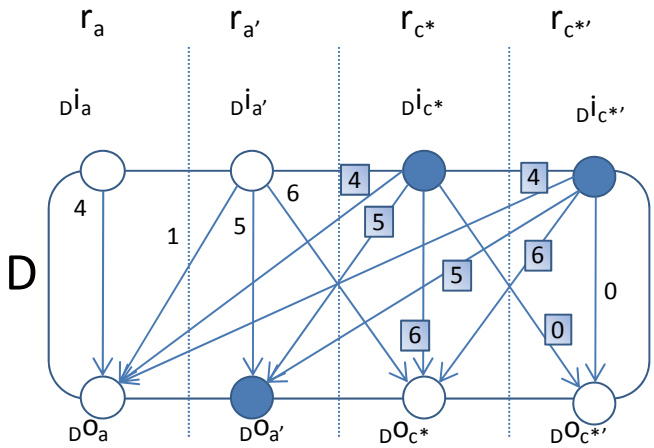
Figure 16a shows an alternative choice between two sub-collaborations A and B. We assume the following: roles a and a' are involved in collaboration A, roles b and b' are involved in collaboration B and there is a distributed choice made by roles c* and c*' for which collaboration is to be executed next, where $c^* \in R(A)$ and $c^{*'} \in R(B)$. This decision making is modeled by the Choice (C) sub-collaboration. None of the roles can start their execution in A or B until the choice is made by both the roles c* and c*'. This is modeled by showing the dependencies from the starting events of role c* and c*' in sub-collaboration C to the ending events of all the roles in collaboration C as shown in Figure 16b. This dependency can affect the delays between starting and ending events of various as new dependencies may have formed between these events.



a – Alternative operator

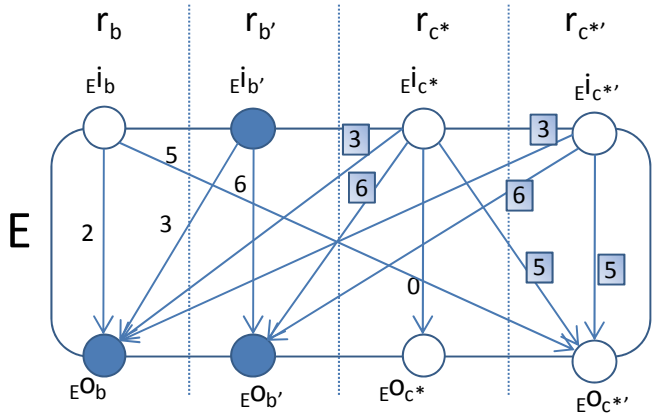


b – CBD of Alternative Operator



c) Resultant Collaboration D if A executes

Figure 16 – Example with a Choice Collaboration



d) Resultant Collaboration E if B executes

Figure 16 – Example with a Choice Collaboration

Figure 16b shows the CBD diagram with the NETDs of sub-collaboration A and B. Let us assume sub-collaboration A is selected for execution in the alternative control flow. Sub-collaboration C weakly sequenced with sub-collaboration A is abstracted by collaboration D in Figure 16c. Delays shown in shaded regions are the delays affected by the weak sequence of A with Choice sub-collaboration. The affects are:

i. There is now a dependency from i_{c*} to o_A. i.e ${}_D i_{c*} \rightarrow {}_D o_a$, with a delay of ${}_D\Delta^{c*}{}_a = 4$. Also there is another dependency from i_{c*} to $o_{c'}*$, even though it has a 0 delay

ii. There is a dependency path from the starting event $i_{c'*}$ to the ending event of all the roles in sub-collaboration A with various delays. Also the delay from ${}_D i_{c*}$ to ${}_D o_{a'}$ is now 5 instead of 2 before.

iii. The delay from i_{c*} to $o_{a'}$, is now ${}_D\Delta^{c*}_{a'} = 5$. Previously, the delay was ${}_A\Delta^{c*}_{a'} = 2$ before weakly sequenced with sub-collaboration C. This is because for $o_{A'}$ to occur, both path of executions ${}_Ci_{c*} \rightarrow {}_Co_{a'} \rightarrow {}_Ai_{a'} \rightarrow {}_Ao_{a'}$ and ${}_Ci_{c*} \rightarrow {}_Co_{c*} \rightarrow {}_Ai_{C*} \rightarrow {}_Ao_{a'}$ must execute. This is satisfied by taking the maximum delay of the two paths: max $(({}_C\Delta^{c*}_{a'} + {}_A\Delta^{a'}_{a'}), ({}_C\Delta^{c*}_{c*} + {}_A\Delta^{c*}_{a'})) = \max(2, 5) = 5$.

iv. Similarly delay from i_{c*} to o_{c*} has changed to ${}_A\Delta^{c*}_{c*} = 6$

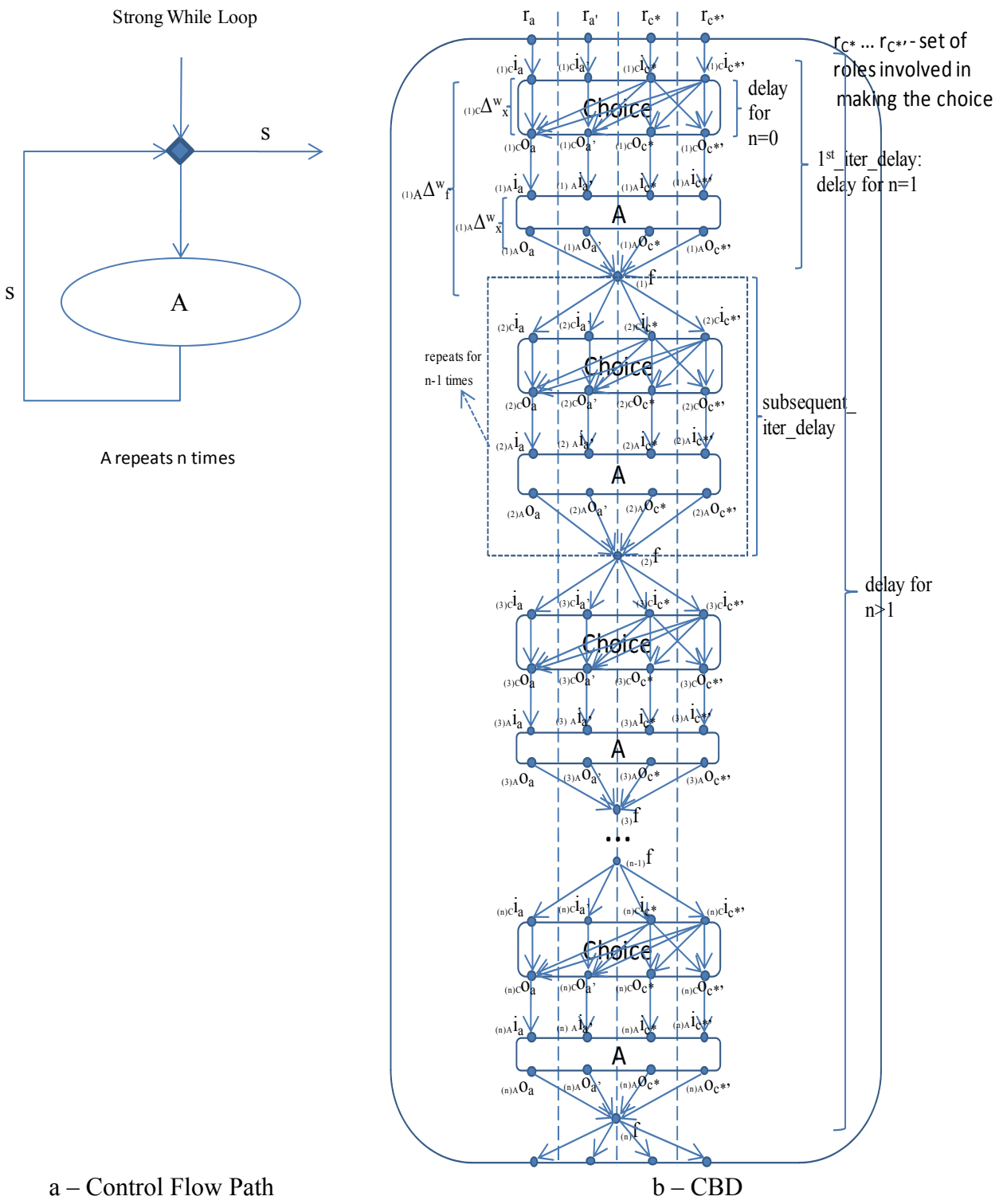
Similar shaded delays are shown in Figure 16d if sub-collaboration B is executed.

5.5 Strict While Loop

[8] defines a "strict while loop" where collaboration C1 is repeated and then followed by C2, as follows: "C_1 is executed zero, one or more times and then C_2 will be executed; more precisely, the behavior starts with a choice between C_1 and C_2; if C_1 is executed, there is **strict** sequencing between the end of C_1 and the choice of executing C_1 again or terminating the loop with C_2." This is written as "C1 *s C2".

Figure 17a shows a control flow diagram of a strict while loop repeating sub-collaboration A. We do not show the follow-up collaboration "B" as the analysis for a sub-collaboration strictly sequenced with B has been previously analyzed in Section 5.1. Hence, we focus on the abstraction of multiple iterations of sub-collaboration A.

Figure 17b shows a partial order diagram that defines the dynamic behaviour of the strict while loop for the control flow path where sub-collaboration A is executed n times. The special case of n=1 is also indicated. In case of n=0, the delays are zero.



a – Control Flow Path b – CBD

Figure 17 – Strict While Loop

To keep track of which iteration is being executed, we introduce the notation ${}_{(j)}\alpha$, where α represents any of the behavioural properties of an collaboration/sub-collaboration and j is an integer representing the number of the iteration. The index j is shown for only those item that relate to different iterations, for example ${}_{(j)A}i$ represents the starting event i of the j^{th} iteration of collaboration A.

The "Choice" mechanism discussed in Section 5.4 is again employed at the beginning of each execution of collaboration A to assess the need for further repetition. Once more, we assume that collaboration A is weakly sequenced after the collaboration Choice. As previously, the *choice* is made by a set C* of roles identified as $r_{c*}...r_{c*'}$.

As this is a strict while loop, all the ending events of each iteration of collaboration A

synchronize at a synchronization event, written ${}_{(j)}s$ for iteration j. However, there is no explicit synchronization before the starting events of the first iteration of collaboration *choice* and A, that is, we do not make any assumption about the kind of sequencing that precedes the composite collaboration D.

5.5.1 Consideration of a Single Control Flow Path

If we consider a single control flow path of a strict while loop where sub-collaboration A repeats n times, the NETD of the composite collaboration D ${}^{(cp)}{}_{D}\Delta^{w}{}_{z}$ for $w \in I(D)$ and $z \in O(D)$ is given by the formulas in Table 8.

Table 8 – Strict While Loop Operator (Single Control Flow Path)

n		**Fixed Delays / Range of Delays**	**Stochastic Delays**
0	if $w \in C^*$	0 (61f)	0 (61s)
	Otherwise	$-\infty$ (62f)	$-\infty$ (62s)
1	if $w \in C^*$	$\max(_{y\,\varepsilon\, I(A),\, x\,\varepsilon\, O(A)}\,(_{(1)}{}^{(cp)}{}_{(mzz)A}\Delta^{y}{}_{x}))$ (63f)	$\prod_{y\varepsilon\, I(A),x\,\varepsilon\, O(A)}\,{}^{(cp)}{}_{(stoc)A}\Delta^{y}{}_{x}(t)$ (63s)
	Otherwise	$\max_{x\,\varepsilon\, O(A)}\,(_{(1)}{}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{x})$ (64f)	$\prod_{x\,\varepsilon\, O(A)}\,{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{x}(t)$ (64s)
> 1	if $w \in C^*$	$\sum_{j=1}^{n}\,(\max_{y\varepsilon\, I(A),\, x\,\varepsilon\, O(A)}\,(_{(j)}{}^{(cp)}{}_{(mzz)A}\Delta^{y}{}_{x}))$ (65f)	$\otimes_{j=1}^{n}\,(\prod_{y\,\varepsilon\, I(A),x\,\varepsilon\, O(A)}\,{}_{(j)}{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{x}(t))$ (65s)
	Otherwise	$\max_{x\,\varepsilon\, O(A)}\,(_{(1)}{}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{x}) + \sum_{j=2}^{n}(\max_{y\varepsilon\, I(A),\, x\varepsilon O(A)}\,{}_{(j)}{}^{(cp)}{}_{A}\Delta^{y}{}_{x})$ (66f)	$(\prod_{x\,\varepsilon\, O(A)}\,{}_{(1)}{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{x}(t)) \otimes \otimes_{j=2}^{n}\,(\prod_{y\,\varepsilon\, I(A),x\,\varepsilon\, O(A)}\,{}_{(j)}{}^{(cp)}{}_{(stoc)A}\Delta^{y}{}_{x}(t))$ (66s)

In the following, we justify the formulas given in Table 8. We assume that all the dependencies shown in Figure 17b by an arrow are associated with a zero delay. As the logic and proof for both stochastic and fixed delay is the same, we only give the proof for fixed delays. Equivalent proofs can be applied to the proofs of stochastic delays

Case n = 0 – Sub-collaboration A executes 0 times, but Choice collaboration executes once

<u>if $w \in C^*$:</u> All the involved roles cannot start their execution until the choice is made by roles $r_{c^*}...r_{c^*}$, in collaboration Choice, causing a dependency from role $r_{c^*}...r_{c^*}$, to all the involved roles.

We had initially mentioned that we assumed that the delay to make the choice and for choice propagation is zero, and hence the delay from the roles $r_{c^*}...r_{c^*}$, to all the involved roles is zero.

otherwise: As there is no execution of A, there is no delay from any starting event to any ending event other than mentioned above. This is represented as $-\infty$ as defined by Equation (12).

Case n = 1 – Sub-collaboration A executes once, but Choice collaboration executes twice

if $w \notin C^*$: Collaboration A executes once, which is equivalent to sub-collaboration A strictly sequenced with a following collaboration. This is calculated partially in Equation (17) in Table 2 as:

$$1^{st}\_iter\_delay = \max_{x \in O(A)} ({}_{(1)}{}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{x}) \quad (67)$$

if $w \in C^*$: If the starting event of role w belongs to a role making the choice, then no execution in collaboration A could have started until this starting event from role *w* occurs. Furthermore, the ending event of role z cannot occur until all the dependencies have been satisfied from all the starting events. Hence the collaboration's delay is the maximum over all the starting and ending events as stated in Equation (67).

Case n > 1: Sub-collaboration A executes n times, and Choice collaboration executes n+1 times

From (63) and (64), we know the delay for A's first iteration.

As seen in Figure 17b, the *subsequent_iter_delay* looks identical to the *1st_iter_delay* with the addition of the synchronization event *f* at the beginning of each iteration, along with the associated dependencies.

When we calculated the delay for 1st_iter_delay above, we assumed that starting events except starting event of role *w* have occurred some time ago. Hence we do not consider the delays from the remaining starting events to the synchronization event $_{(1)}f$.

This is not the case for the 2nd iteration and onwards, as the starting events for those iterations cannot occur until $_{(n-1)}f$ has occurred. Only then, the execution of the nth iteration may start. Since the delay from all the starting events need to be considered, applying (6) to (67), we get:

$$subsequent\_iter\_delay = \max_{w \in I(A)} (\max_{x \in O(A)} ({}_{(j)}{}^{(cp)}{}_{A}\Delta^{w}{}_{x})) \quad (68)$$

For each of these iterations, the NETD depends on the execution path of the body of A being executed. Hence, this delay could be different for each iteration depending on the control flow path in A. The total delay can be calculated as:

$$2\_to\_n\_delay = \sum_{j=1}^{n} (\max_{w \in I(A)} (\max_{x \in O(A)} ({}_{(j)}{}^{(cp)}{}_{A}\Delta^{w}{}_{x}))) \quad (69)$$

From Figure 17b, it is quite evident that the NETD of the composite collaboration is the sum of *1st_iter_delay* and *2_to_n_delay*, which gives (66).

5.5.2 Consideration of Multiple Control Flow Paths – Range of Delays

If we consider all the possible control flow paths, then based on the formulas from Table 8, it is quite evident that the minimum delay occurs for n = 0 and the maximum delay occurs when n = ∞. This leads to:

Minimum Delay: $^{(cp)}{}_{(min)D}\Delta^{w}{}_{z} = 0$ if $w \in C^*$ (70a)

$^{(cp)}{}_{(min)D}\Delta^{w}{}_{z} = -\infty$ otherwise (70b)

Maximum Delay:

$$^{(cp)}{}_{(max)D}\Delta^{w}{}_{z} = \infty \quad (71)$$

5.5.3 Consideration of Multiple Control Flow Paths – Stochastic Delays

For stochastic delays, it is assumed that each time the Choice is performed, there is a probability p that A is executed and a probability *q=1-p* to stop the iterations. Then the NETD for collaboration D, $^{(cp)}{}_{(stoc)D}\Delta^{w}{}_{z}$, is given by:

if $w \in C^*$: $$q * \sum_{i=1}^{n} p^{i} \otimes_{j=1}^{i} {}^{(j)}{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{z} \quad (72)$$

Otherwise: $$pq * \prod_{x \varepsilon O(A)} {}^{(1)}{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{x}(t) \otimes \quad (73)$$

$$(\delta + \sum_{i=1}^{n-1} p^{i} \otimes_{j=1}^{i} \prod_{y \varepsilon I(A), x \varepsilon O(A)} {}^{(j)}{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{x}(t))$$

and there is the probability q that there is no dependency from w to z, when w $\notin$ C*. This happens when n = 0.

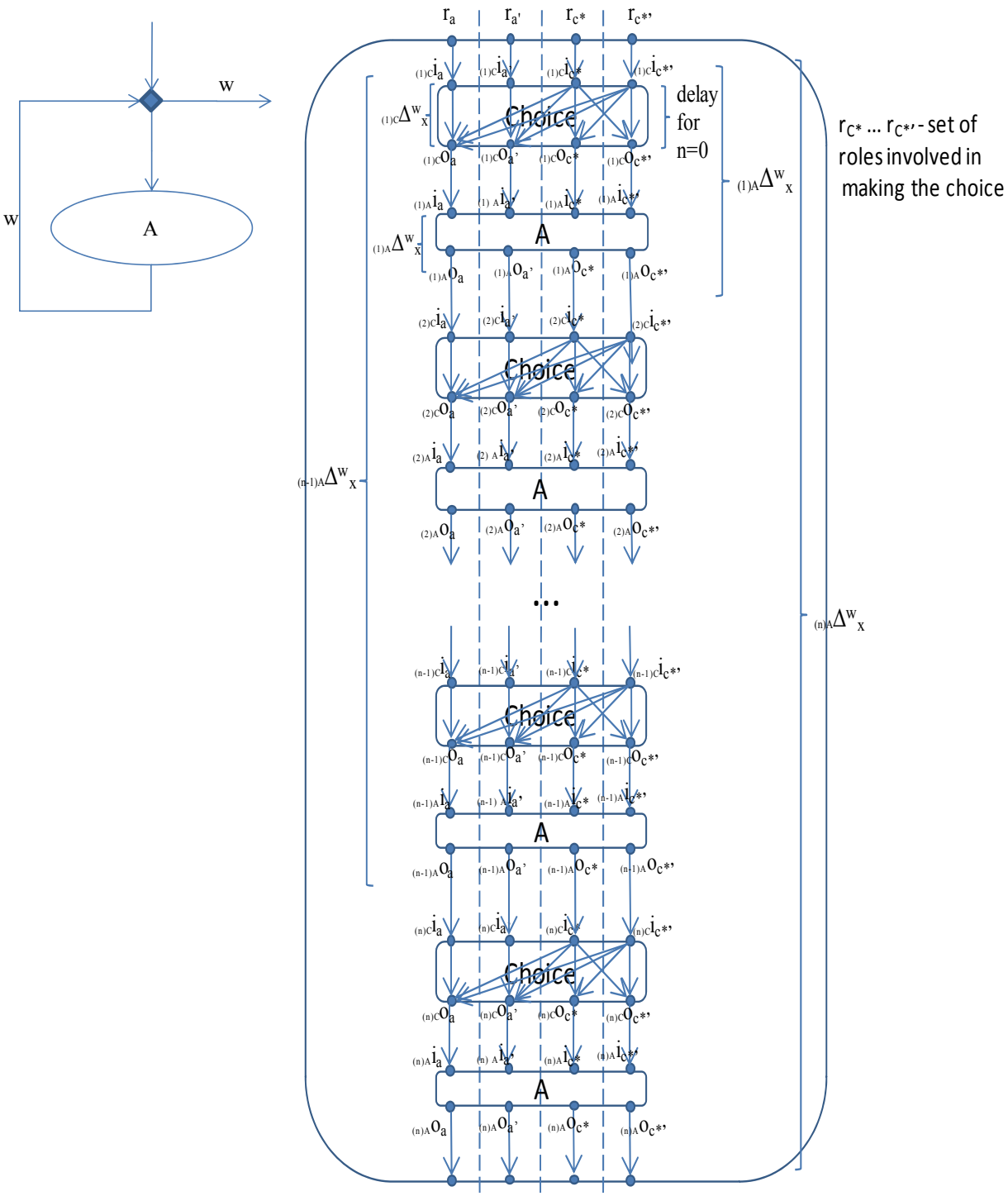
Discussion: The above delay is calculated by applying Equation (7) to the formulas in Table 8. We apply probability p for each time A executes and probability q once for A to stop the iteration and we sum up these delays.

if w $\in$ C*: If the starting event belongs to the roles involved in making the Choice, then applying (7) to (61s), (63s) and (65s) and summing up the delays yields (72).

Otherwise: If the starting event does not belong to a role involved in making the Choice, then applying (7) to (62s), (64s) and (66s) and summing up the delays yields (73). If the sub-collaboration does not iterate at all (i.e. n = 0), then there obviously is no delay from w to z and hence no dependency from w to z.

5.6 Weak While Loop

[8] defines a weak while loop with body C_1 followed by collaboration C_2 similar to that of a strict while loop "...except that **weak** sequencing is used between the end of C_1 and the choice of executing C_1 again or terminating the loop with C_2." This is annotated as C1 *w C2. Figure 18 shows such a weak while loop and its CBD definition.



a) Control Flow b) CBD

Figure 18 – Weak While Loop

5.6.1 Consideration of a Single Control Flow Path – Fixed Delays

NETD: If we consider a single control flow path of a weak while loop where sub-collaboration A repeats n times, the NETD for w ∈ I(A) and z ∈ O(A) for the composite collaboration D, ${}^{(cp)}{}_{D}\Delta^{w}{}_{z}$, is given in Table 9.

Table 9 – Weak While Loop

n		**Fixed Delays / Range of Delays**	**Stochastic Delays**
0	if w ∈ C*	0 (74f)	0 (74s)
	otherwise	$-\infty$ (75f)	$-\infty$ (75s)
1	if w ∈ C*	$\max_{y\in I(A)} ({}_{(1)}{}^{(cp)}{}_{(mzz)\,A}\Delta^{y}{}_{z})$ (76f)	$\prod_{y\varepsilon I(A)}({}_{(1)}{}^{(cp)}{}_{(stoc)\,A}\Delta^{y}{}_{z})$ (76s)
	otherwise	${}_{(1)}{}^{(cp)}{}_{(mzz)A}\Delta^{w}{}_{z}$ (77f)	${}_{(1)}{}^{(cp)}{}_{(stoc)A}\Delta^{w}{}_{z}(t)$ (77s)
> 1		$\max_{x\in O(A)} ({}_{(n-1)}{}^{(cp)}{}_{(mzz)\,D}\Delta^{w}{}_{x} + {}_{(n)}{}^{(cp)}{}_{(mzz)A}\Delta^{x}{}_{z})$ (78f)	$\prod_{x\varepsilon O(A)}({}_{(n-1)}{}^{(cp)}{}_{(stoc)D}\Delta^{w}{}_{x} \otimes {}_{(n)}{}^{(cp)}{}_{(stoc)A}\Delta^{x}{}_{z})$ (78s)

To justify the formulas given in Table 9, we assume that all the dependencies shown in Figure 18b by an arrow are associated with a zero delay.

Case n = 0:

When A iterates 0 times, then similar to strict while loop, there is no delay as we assumed that the process of making the choice and its propagation this choice occurs with zero delay.

Furthermore, for the starting and ending events of the same roles, there is zero delay as there is no execution of A.

As none of the roles can start their executions in A until the starting event in C* have occurred, we have a dependency from roles belonging to C* to all the roles, as shown by a zero delay from the starting event of C* to the ending events of all the roles in A. For the remaining starting and ending events, as A has not executed, there is no dependency, and hence the delay is $-\infty$ (because of Equation 12).

Case n =1:

The proof for (74f) and (75f) is similar to the proof of Equations (61f) and (62f) of Table 8. However, none of the roles may start their execution in A until the choice is made by the roles in C*. For the ending event of z to occur, dependencies from all the starting events of A to the ending event of z must occur.

If w belongs to C*, then dependencies from the starting events of all the roles to the ending event of z must be satisfied and therefore we consider delays from the starting events of all the roles to the ending event of the desired role in A.

If w does not belong to C*, the only dependency remaining is from the starting event of w to the ending event of z and hence we only consider the delay between these two events.

Case n > 1 :

This is calculated using a recursive formula where ${}_{(n)}{}^{(cp)}{}_{D}\Delta^{w}{}_{z}$ denotes the NETD for the n^{th} iteration of the composite collaboration D.

As can be seen in Figure 18b, the (n-1) first iterations of collaboration A can be abstracted by ${}_{(n-1)}{}^{(cp)}{}_{D}\Delta^{w}{}_{x}$. To calculate the n^{th} iteration of collaboration A, we consider first the previous (n-1) iterations with the delay ${}_{(n-1)}{}^{(cp)}{}_{D}\Delta^{w}{}_{x}$ which are weakly sequenced with the n^{th} iteration with the delays ${}_{(n)}{}^{(cp)}{}_{D}\Delta^{w}{}_{x}$. The formula for the weak sequencing of these delays is discussed in Section 5.2.

5.6.2 Consideration of Multiple Control Flow Paths – Range of Delays

We consider multiple control flow paths of a weak while loop where sub-collaboration A repeats n times, $0 \leq n \leq \infty$. Based on Table 9, the minimum delay is either zero (if the starting event belongs to a role making the choice) or $-\infty$ for the remaining roles. This happens when n = 0. The maximum delay is ∞ when $n = \infty$:

Minimum Delay:

$${}^{(cp)}{}_{(min)D}\Delta^{w}{}_{z} = 0 \qquad \text{if } w \in C^{*} \tag{79}$$

$${}^{(cp)}{}_{(min)D}\Delta^{w}{}_{z} = -\infty \qquad \text{otherwise} \tag{80}$$

Maximum Delay: $^{(cp)}{}_{(max)D}\Delta^{w}{}_{z} = \infty$ (81)

5.6.3 Consideration of Multiple Control Flow Paths – Stochastic Delays

We consider all the possible control flow paths and assume that each leads with probability p to the execution of A and probability *q=(1-p)* to the termination of the loop. Based on the formulas for fixed delays, the NETD for collaboration D, $^{(cp)}{}_{(stoc)D}\Delta^{w}{}_{z}$ can be calculated as:

$$q \sum_{i=1}^{n} p^{i} * {}_{(i)}{}^{(cp)}{}_{(stoc)D}\Delta^{w}{}_{z} \quad (82)$$

and there is the probability q that there is no dependency from w to z, when $w \notin C^*$. This happens when n = 0.

The proof is very similar to the proof of (72) and (73). If the role w does not belong to C*, there exists a probability q for no dependency from w to z. This can happen when n=0.

5.7 Summary

In this chapter, we analyze performance of global collaborations based on the performance of the constituent sub-collaborations and the sequencing operators. We consider composite collaborations, composed of two sub-collaborations with various sequential operators such as weak sequence, strict sequence, concurrency, alternatives, weak while loop and strict while loop. We classify roles according to their participation in these sub-collaborations. Based on these role classifications, we proposed formulas to calculate fixed, stochastic and range of delays for single and multiple control flow paths.

6. APPROXIMATION WITH NORMAL DISTRIBUTIONS

Based on the formulas of Chapter 5, we can analyze performance of global collaborations based on the delays of the constituent sub-collaborations. These formulas have been provided for deterministic delays as well as for delays that are of stochastic nature.

As many natural phenomena can be approximated very well by a normal distribution, such distributions have been developed into a standard of reference for many probability problems. Another reason for their popularity is the central limit theorem, which states that under certain conditions, the mean of a large number of independent, identically distributed random variables is distributed approximately normal, regardless of the form of underlying distribution. Hence, the sum of independent processes often have a distribution very close to normal.

For these reason, in this chapter, we would like to analyze performance of global collaborations based on normally distributed delays of sub-collaborations and use approximations as much as deemed appropriate.

6.1 Normal Distribution

A normal distribution is a continuous probability distribution, defined by the probability density function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2(\sigma^2)}} \tag{83}$$

where μ is the mean/expectation of the distribution, σ is the standard deviation and hence σ^2 is the variance.

As can be seen from Figure 19, the probability density function (PDF) of normal distributions is symmetric with a single central peak at its mean. The shape of the PDF curve is described as bell-shaped with the graph falling off evenly on either side of the mean – fifty percent of the distribution lies to the left of the mean and fifty percent lies to the right of the

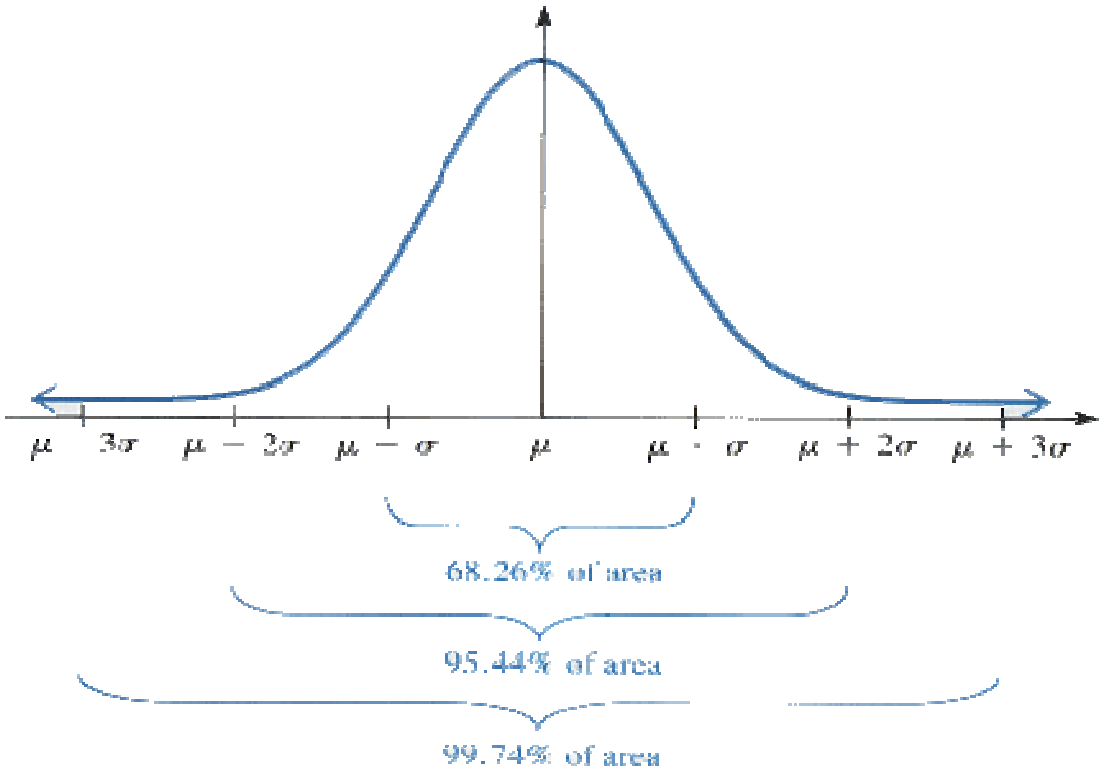


Figure 19 – Probability Density Function of a Normal Distribution [31]

mean. The spread of a normal distribution is dictated by the standard deviation – the smaller the standard deviation the more concentrated the data. It is continuous for all values of X between –

∞ and +∞ such that each conceivable interval of real numbers has a probability different than zero. About 2/3 space (68.26%) of all cases fall within one standard deviation of the mean and about 95% of all cases lie within two standard deviations. The simplest case of a normal distribution is called a standard normal distribution when $\mu = 0$ and $\sigma = 1$.

6.2 Algebra with Normal Distributions

For the following, unless explicitly stated otherwise, we assume all the distributions are independent and hence have independent random variables which are normally distributed.

6.2.1 Normal Random Variables in Sequence

Figure 7a shows events in sequences. If these events have delays defined as normally distributed random variables, then the sum of these delays can be defined by the convolution of their individual distributions. If X_1, $X_{2...}$ X_n are mutually independent random normal variables with means μ_1, μ_2,…,μ_n and variances σ_1^2, σ_2^2… σ_n^2, then the sum of these random variables results in a normal distribution [34]:

$$\sum_{i=1}^{n} \text{Normal}(\mu_i, \sigma_i^2) = \text{Normal}\left(\sum_{i=1}^{n} \mu_i, \sum_{i=1}^{n} \sigma_i^2\right) \qquad -\infty < \mu_i < \infty, \quad \sigma_i^2 > 0 \qquad (84)$$

This states that the sum of independent random variables is normal with its mean being the sum of the constituent means and its variance being the sum of its constituent's variances.

6.2.2 Maximum/Minimum of Normally Distributed Random Variables

There are cases when taking a maximum or a minimum of normally distributed random variables is required such as in various formulas mentioned in Chapter 5. We can calculate the maximum and the minimum for a general distribution delay by Equations (8) and (9).

Figure 20 illustrates the PDF of two normally distributed random variables $X_1(\mu_1, \sigma_1^2)$ and $X_2(\mu_2, \sigma_2^2)$. From Equations (8) and (9), we know that the maximum/minimum of normal distributions requires the product of normal distributions, which unfortunately does not result in a normal distribution. Hence for this reason, we propose an approximation of the product of two normal distributions as a normal distribution.

We base our approximations on the *distance* between the means of the two PDFs. In other words we are going to present approximations for a *little overlap* or a *strong overlap* of the curve of the PDF of two distributions.

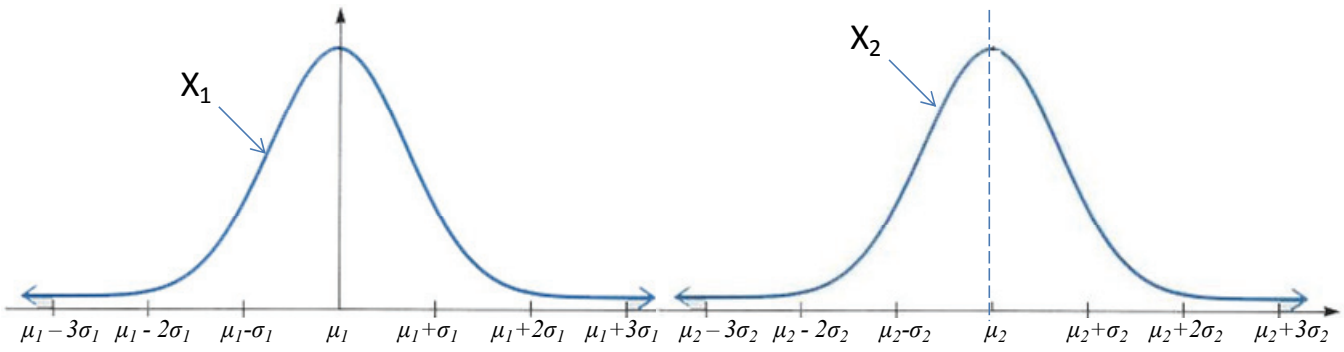


Figure 20 – PDF of Normal Distribution of X_1 and X_2

6.2.2.1 Little Overlap

Let us examine distributions X_1~N(0,1) and X_2~N(4,1) in Figure 21. We say that the *distance* can be measured between the mean of the two distributions by the number of standard deviations away (explained below). The standard deviation is a good candidate as it measures how spread out a distribution is around the mean.

Let us assume there is a difference of *m* standard deviations between the two means i.e. $(\mu_2 - m\sigma_2) > (\mu_1 + m\sigma_1)$ or $(\mu_2 + m\sigma_2) < (\mu_1 - m\sigma_1)$. If the mean of X_1 is *sufficiently* greater than the sum of mean and standard deviations of both distributions, then the maximum distribution can be approximated by X_1 and the minimum distribution by X_2. Depending on this multiple *m*,

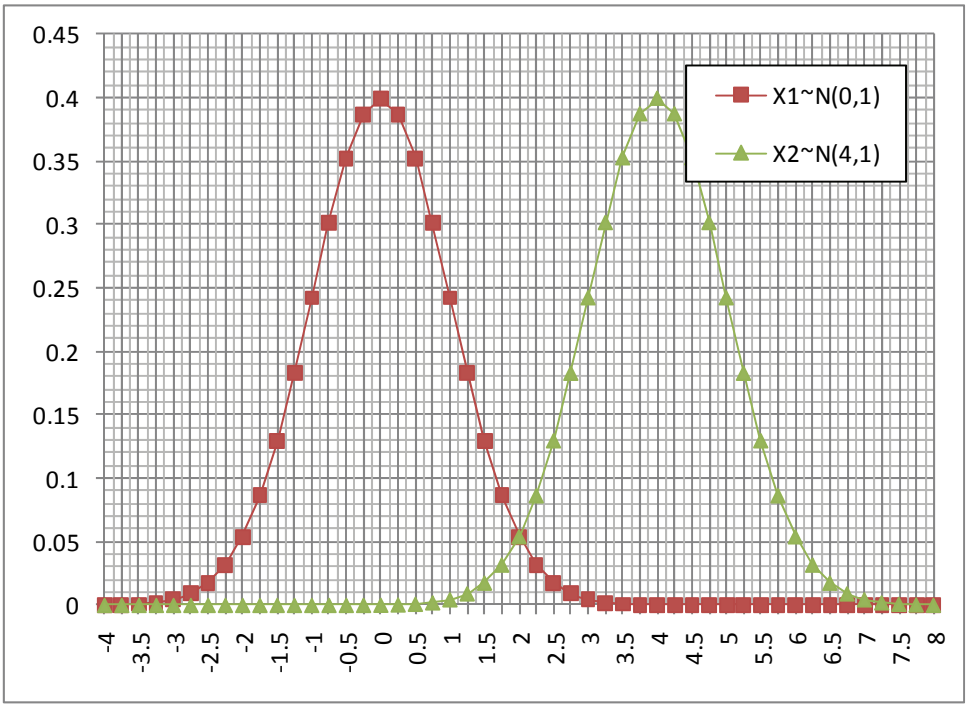


Figure 21a – PDF of X_1~N(0,1) and X_2~N(4,1)

we can say how good this approximation is. For example, if m = 3 (or higher), then we can say that this is a good approximation as we know that approximately 99.7% (or more) of the data lies within this 3 (or more) standard deviations of the mean and does not overlap with the other distribution. If m = 2, then most of the data lies within 95% and hence only 5 % data overlaps and if m=1, then 68% data lies around the mean and 32% data overlaps, and therefore the approximation is not good.

In Figure 21, X_1 has a mean of 0 and standard deviation of 1 while X_2 has a mean of 4 and standard deviation of 1. We can see that these two distributions are quite far away from each

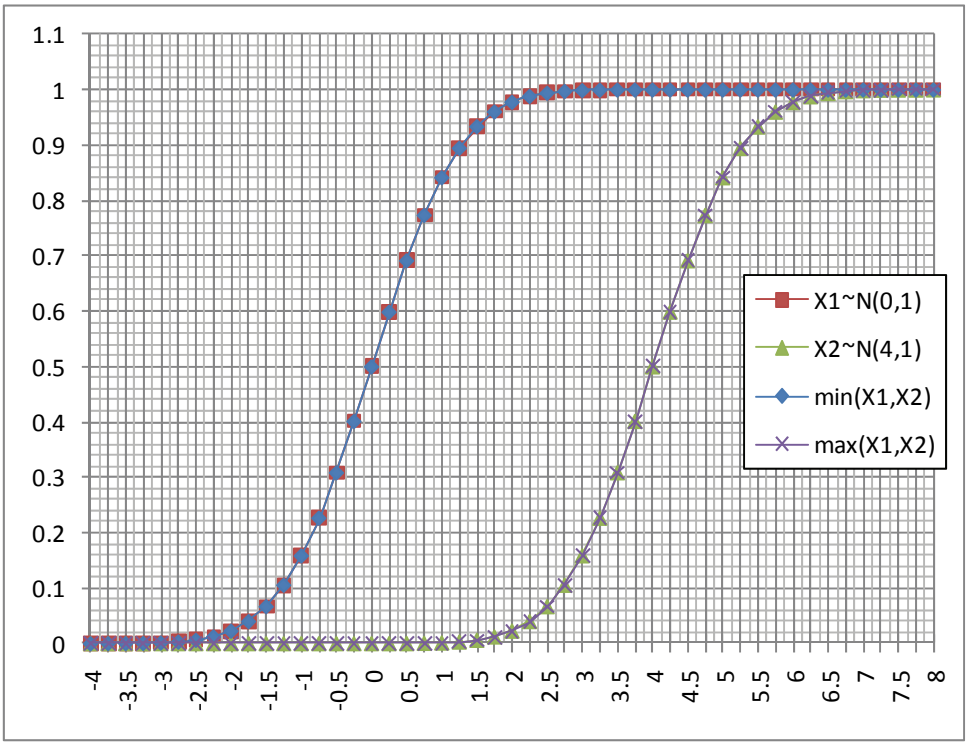


Figure 21b – CDF of X_1, X_2, Minimum and Maximum

other – they are 4 standard deviations away from each other's mean. Using the functions provided in Microsoft Excel, we tabulated the PDF and CDF of X_1 and X_2. Using the Equations (8) and (9), we also calculated the distributions for the maximum and the minimum curve. These values are provided in Table 22 in Annex C.1. Based on these values, we graph the curves for the maximum and minimum as seen in Figure 21b. We can clearly see that the curve for the minimum follows the curve for the X_1~N(0,1) distribution and the curve for the maximum follows the curve for X_2~N(4,1), as expected.

6.2.2.2 Strong Overlap

If there is a *strong* overlap between the two distributions, as in the case of X_1~N(1.5,1.5) and X_2~N(1,1) as shown in Figure 22, we can still approximate the maximum and minimum under certain conditions.

If the means of two distributions, X_1 and X_2, have the same value (assumed to be zero in the following) with variances σ_1^2 and σ_2^2, respectively, then the distribution of the product of X_1 and X_2 can be calculated as follows [85]:

$$F_{XY}(u) = \iint_{-\infty}^{\infty} \frac{e^{-x^2/(2\sigma_1^2)}}{\sigma_1\sqrt{2\pi}} \frac{e^{-y^2/(2\sigma_2^2)}}{\sigma_2\sqrt{2\pi}} \delta(x\,y-u)\,dx\,dy \quad (85)$$

$$= \frac{K_0\left(\frac{|u|}{\sigma_1\sigma_2}\right)}{\pi\,\sigma_1\sigma_2} \quad (86)$$

where $\delta(x)$ is the Dirac delta function and K_0 is a modified Bessel function of the second kind.

He et al. [32] state that if the mean has the same value as the variance for each variable, then the exact solution of the CDF, PDF, mean and standard deviation of Z = min{X,Y} can be calculated as follows:

CDF: $F_z(z) = 1 - P\{\min(X_1,X_2) > z\}$ (87)

$= 1 - P\{X_1 > z, X_2 > z\}$ (88)

$= 1 - [1 - X_1(z)]\,[1 - X_2(z)]$ (89)

$= F_{X1}(z) + F_{X2}(z) - F_{X1}(z)F_{X2}(z)$ (90)

PDF: $f_Z(z) = f_{X1}(z) + f_{X2}(z) - f_{X1}(z)\,F_{X2}(z) - f_{X2}(z)F_{X1}(z)$ (91)

Mean: $\mu_Z = \mu_{X1} + \mu_{X2} - \int_{-\infty}^{\infty} z\, f_{X1}(z)\, F_{X2}(z)\, dz - \int_{-\infty}^{\infty} z\, f_{X2}(z)\, F_{X1}(z)\, dz$ (92)

Variance: $\sigma_Z^2 = \int_{-\infty}^{\infty} (z - \mu_Z)^2 f_Z(z)\, dz$ (93)

As the above formulas requires extensive computation, [32] provides further approximations to the mean and variance.

$\mu_Z \approx \mu_{min} + b_1(\mu_{min})^{1/2}\, e^{b_2(\mu_{min})^{b_3}\delta}$ (94)

$\sigma_Z^2 \approx \Delta_{\sigma^2}^{*}\,(1 - e^{-(\frac{\mu_Z - \mu_{min}}{h_1})^{h_2}}) + (\sigma_Z^2)^{*}$ (95)

where $h_1 = c_1(\mu_{min})^{c_2}$, $h_2 = c_3(\mu_{min})^{c_4} + c_5$,

$c_1 = 1.911$, $c_2 = 0.4986$, $c_3 = -1.672$, $c_4 = -0.4177$, and $c_5 = 1.996$,

$\mu_{min} = \min(\mu_1, \mu_2)$ and $\delta = |\mu_1 - \mu_2|$

For further details on (94) and (95), please refer to [32].

Equations (94) and (95) yields the approximation of the mean of the minimum of X_1~N(1.5,1.5) and X_2~N(1,1) as 0.6085.

For these distributions, X_1~N(1.5,1.5) and X_2~N(1,1), we have calculated the CDF of the minimum curve using the Equations (8) and (9) in Microsoft Excel. The results are graphed in Figure 22b and presented in Table 23 of Annex C.2. We can see that the mean of the minimum curve is between 0.5 and 0.65, while Equation (94) of the approximation method of [32] yielded 0.6085. This gives us confidence in the approximation using Equations (94) and (95).

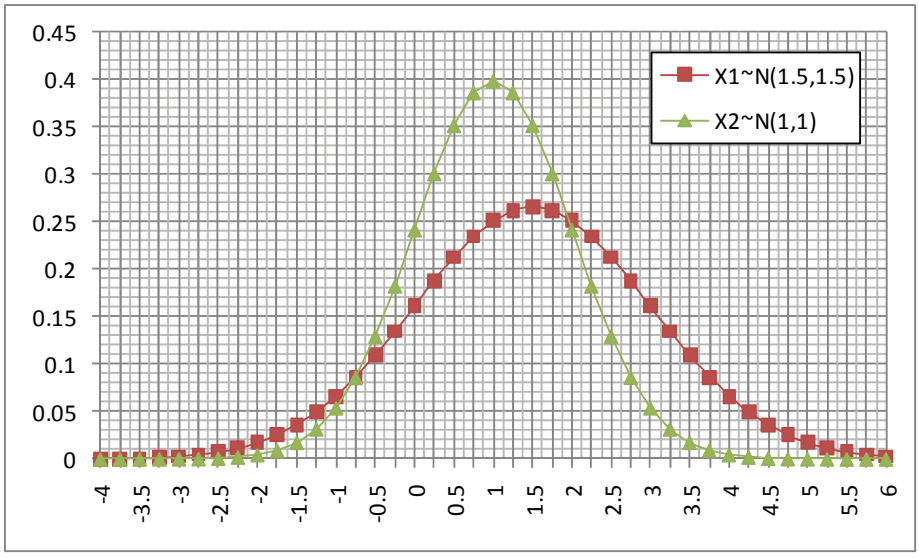


Figure 22a - PDF of X_1~N(1.5,1.5) and X_2~N(1,1)

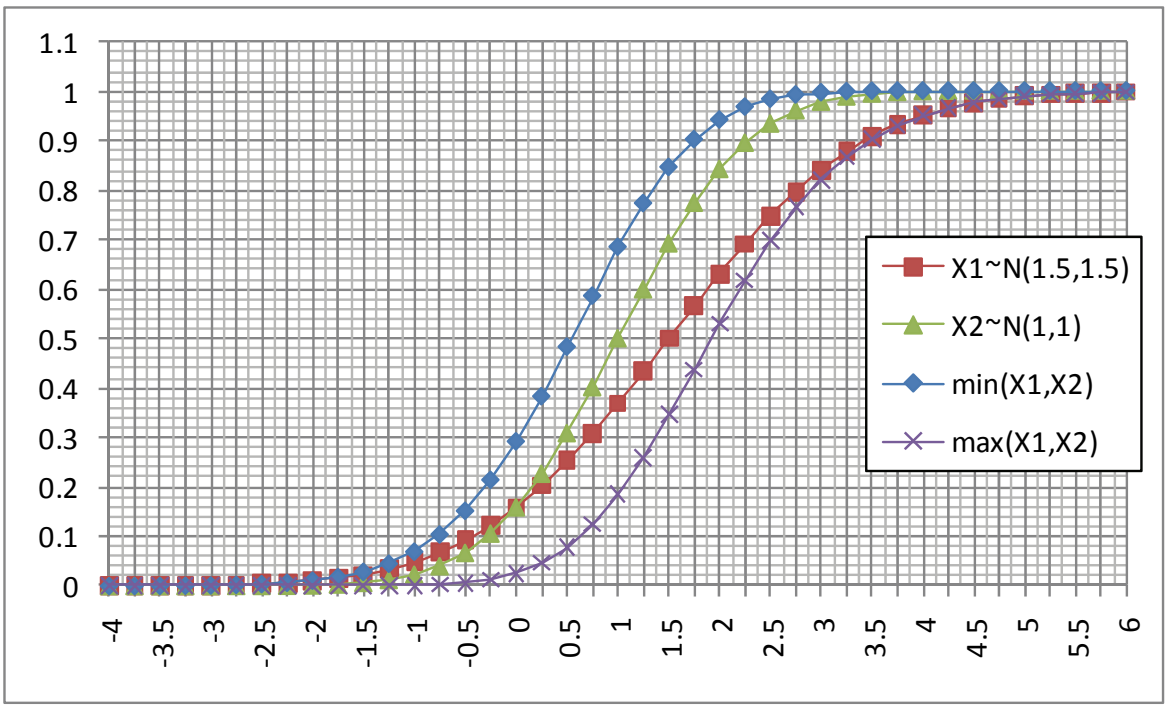


Figure 22b – CDF of X_1, X_2, Minimum and Maximum

6.3 Summary

In this chapter, we approximated the analysis of stochastic delays of Chapter 5 for Normal Distributions. Normal Distributions are important since the mean of a large number of independent, identically distributed random variables is distributed approximately normal, regardless of the form of underlying distribution. We analyze normal distributions in a sequence. We also consider concurrency and distinguish the cases of "little overlap" and "strong overlap" of the PDFs of the given distributions; we also provide formulas to calculate the maximum or the minimum between two concurrent normal distributions.

7. DISCUSION OF ASSUMPTIONS

Our work relies on certain assumptions made to analyze the performance of composite collaborations. In this chapter, we discuss some of their implications and try to estimate the amount of error that may be caused by these assumptions.

7.1 Shared Resources

Shared resources such as CPUs, processor caches, physical memory, or disk access, typically refer to a hardware or a software resource that can be accessed by one or more processes. As shared resources increase the complexity of a performance model, recent work on performance modeling often tends to omit shared resources [25, 38, 42, 49, 50] or emphasize the high cost of solving performance model when shared resources are involved [38].

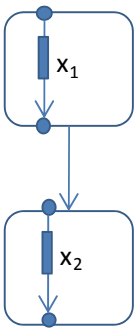
7.1.1 Multiple Simultaneous Users of a Single Shared Resource

So far in our work, we have assumed that there is no resource sharing. While this may intrinsically simplify our analysis, this is far from the reality. Let us consider several concurrent users using a system, each requesting similar if not the same service.

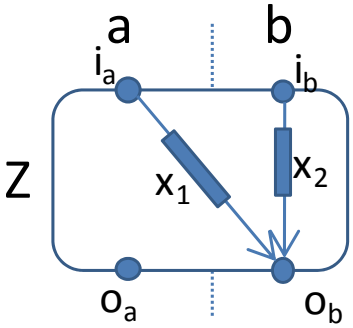
For instance suppose that there is only one resource, which is a CPU accessed simultaneously by several applications. If there is *fair* timesharing, and assuming the performance of these applications are not influenced by any other factors such as memory paging, disks read/write, then we can say that the time delay of actions executed on the CPU has a direct relationship with the number of users accessing the CPU – n applications sharing the

same CPU will run *n* times slower than they *normally* would, that is, when they run alone. Hence, they all slow down by a factor of *n*. If we consider delays to be fixed delays and if we know this factor *n*, then we can calculate the new fixed delays by multiplying the *normal* fixed delays by the factor of *n*.

For stochastic delays, suppose there is only one application accessing the CPU, and the CPU is also used for other things. The speed of the CPU depends on the load. If we run this particular application, the time to execute all the action in this application will be affected by some factor, making it larger or smaller depending on the load. If we repeat this experiment several times, then we will get a stochastic behaviour (this also gives rise to dependent stochastic distributions discussed later in Section 7.2).



a – actions in sequence



b – Concurrent Executions

Figure 23 – Execution of Actions x_1 and x_2

This is illustrated in Figure 23b, where there are two actions x_1 and x_2. If x_1 and x_2 are executed on independent CPUs, then the performance analysis is as described in Chapter 6. However, if x_1 and x_2 are executed on the same processor, then we can analyze the two extreme

cases, depending on when, relative to one another, x_1 and x_2 execute. If the execution of x_1 is completed before x_2 starts (or vice versa), then there are no effects and the performance is as if x_1 and x2 were executed on independent processors. On the other hand, if x_1 and x_2 run as separate threads in the same application, then it will take twice as much time as it would take on independent processors. This is of course assuming fair scheduling of the CPU, and x_1 and x_2 are only utilizing one resource, CPU, and there is no other factor (such as disk read/write, or memory paging) which may influence performance of these two actions.

7.1.2 Other Scenarios with Sharing of Resources

Suppose, there is a single user in a system, but who has initiated concurrent actions, which use a common resource that is not a CPU, but rather a disk or a database, or a CPU that is highly utilized. With these resources, other delays may exist due to queuing, and waiting times, which have not been accounted for in our performance analysis of Section 5.

Consider the two actions x_1 and x_2 in Figure 23b. Let us assume that x_1 and x_2, both, have a *write* action on the same database. We can measure the delay of x_1 and x_2 as per our testing method from Section 4.1. This delay, let us call it *normal delay*, will be the delay of x_1 and x_2 provided that x_1 and x_2 do not run simultaneously. Now, let x_2 execute and during x_2's execution, allow x_1 to execute also and vice-versa. This will give a new delay for $x_1$and x_2, let's call it "*delay with shared resource"*. This "*delay with shared resource"* would typically be larger than the *normal delay* due to sharing the access of the database. If there are many competing processes, consideration needs to be given to additional parameters such as queuing

time, service time, think time, arrival rate, etc. If we know these performance attributes, then modeling paradigms such as Queuing Models can be used as a layer below to calculate the delays for each activity. This allows us to predict delays with shared resources.

7.2 Dependence and Independence

In probability theory, two events are said to be independent if the outcome of one event does not affect the outcome of the second event. For example rolling a die does not have any effect on tossing a coin. So far, in our work, we had assumed that the probabilities involved in our analysis of Section 6 and the stochastic distributions are independent of one another. In this section, we will discuss what impact dependencies can have on our performance analysis.

7.2.1 Dependent Probabilities

We use probabilities in the analysis of alternatives and repetitions of weak and strict while loops in Section 6. In the alternative operator, a probability is introduced which determines how likely a path of execution is amongst several paths of executions, while in strict and weak while loops, the probability of repeating the execution of the body of the loop was introduced. These probabilities had been assumed to be independent.

Let us assume that such probabilities are not always independent. For example, if we consider a loop, let us assume that there is a higher probability for the body of loop to iterate if the body of the loop has already been iterated. This is analogous to a student writing an exam: If the student knows the answer to a question and he knows that the answer is correct, then the

student's confidence increases and the probability for him to get the next question correct also increases.

Another example is a loop for selecting a winning ball out of a given set of initially n balls. Here the probability to terminate the loop increases every time the loop is entered. The probability of selecting the winning ball initially is 1/n, and at the n^{th} repetition, it is one.

To estimate the effects of dependent probability, one needs to know the details of the behaviour, a measure of the correlation, and then perhaps one could give a better approximation for these dependent probabilities.

7.2.2 Dependent Distributions

In our work, we have assumed that the stochastic delays of different sub-collaborations are independent and also the Nominal Execution Time Delays (NETDs) of a given sub-collaboration, from Section 5, are independent of one another. Let us consider cases where these delays are not independent.

Sequence:

In Section 3.1.4.1, we presented Equation (1) to calculate the resultant distribution of two independent random variables in series. However, if these random variables represent a dependent distribution, then Equation (1) and its analysis must be replaced by the following:

If we assume that two continous random variables X and Y represent dependent distributions, and we know the joint probability distribution function (PDF), $f_{X,Y}(x,y)$, of X and Y, then we can calculate the cumulative distribution function (CDF) of Z=X+Y by [28]:

$$F_Z = P(Z \leq z) = P(X + Y \leq z) = \int_{-\infty}^{\infty} \int_{-\infty}^{z-y} f_{X,Y}(x,y)\, dx\, dy$$

and differentiating the above equation will result in the following PDF:

$$f_Z(z) = dF_Z(z)/dz = \int_{-\infty}^{\infty} f_{X,Y}(z-y,y)\, dy$$

These equations can be used to calculate the PDF and CDF of the cumulative distribution. Let us consider the sequence of actions x_1 and x_2 of Figure 23a. If we consider x_1 and x_2 to have two delta distributions such as shown in Figure 23b, where there is a probability of 0.5 for the value to be 10 and probability of 0.5 for the value 20. The average of the distribution is 0.5*10 + 0.5 *20 = 15. If x_1 and x_2 are independent distributions, then the distribution of the delay for both actions in sequence is as follows: probability of 0.5 for the values to be 10 and 20 and probability of 0.5 for the values to be 10 and 20, and the average is 15 + 15 = 30. If x_1 and x_2 are 100 % dependent distributions (to examine the extreme case, we consider complete dependency - where x_1 and x_2 always have the same value), then the distribution of the delay for both actions is 10 + 10 = 20 and 20 + 20 = 40 and the average is (20 + 40) / 2 = 30. This corresponds to the linearity property of the expected value operator [77]:

$$E[X+Y] = E[X] + E[Y]$$

Concurrency:

For calculating the delay of concurrent actions, one needs to take the maximum of two individual delays. In Section 3.1.4.2, we calculated the CDF of the maximum of two independent distributions by taking the product of the CDFs of these two distributions. For dependent distributions, while we were unsuccessful at an analystical solution, we are still able to discuss the average of the maximum of the two individual delays.

If the individual delays have identical distributions which are completely dependent, then the average of the maximum delays is the same as the average of the individual distribution. However, if the individul delays have independent distributions, then the average of the maximum distribution is not the average of the individual delays, but is greater than the average of the individual distributions.

Let us consider again the actions x_1 and x_2 of Figure 23b. We consider x_1 and x_2 to have delta distributions such as shown in Figure 23b, where there is a probability of 0.5 for the value to be 10 and probability of 0.5 for the value to 20. If these delays are assumed to be independent, then the average of the maximum can be calculated by 0.25 * 20 + 0.25 * 20 + 0.25 *20 + 0.25 * 10 = 17.5. However, if these delays are assumed to be completely dependent, the distribution of the maximum will be the same as the distribution of each original delay, and therefore the average of the maximum would be (10+20)/2 = 15.

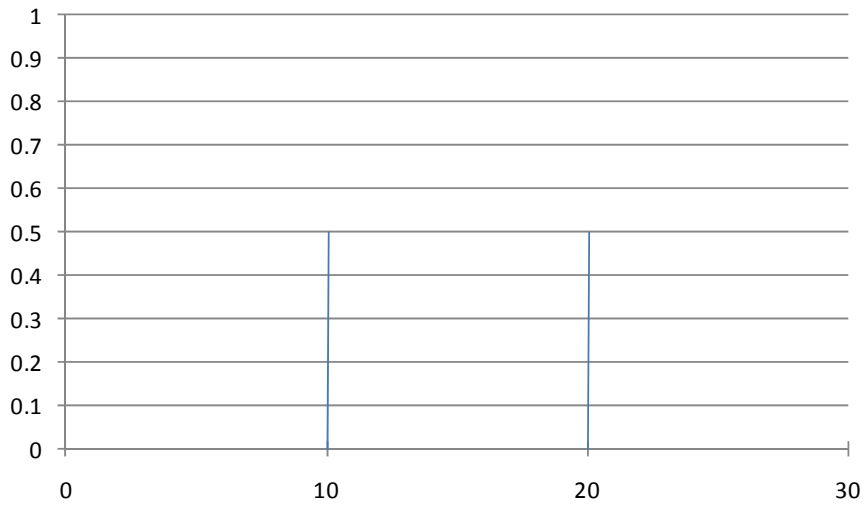


Figure 23b – Dirac Delta Distributions of x_1 and x_2

7.3 Messages

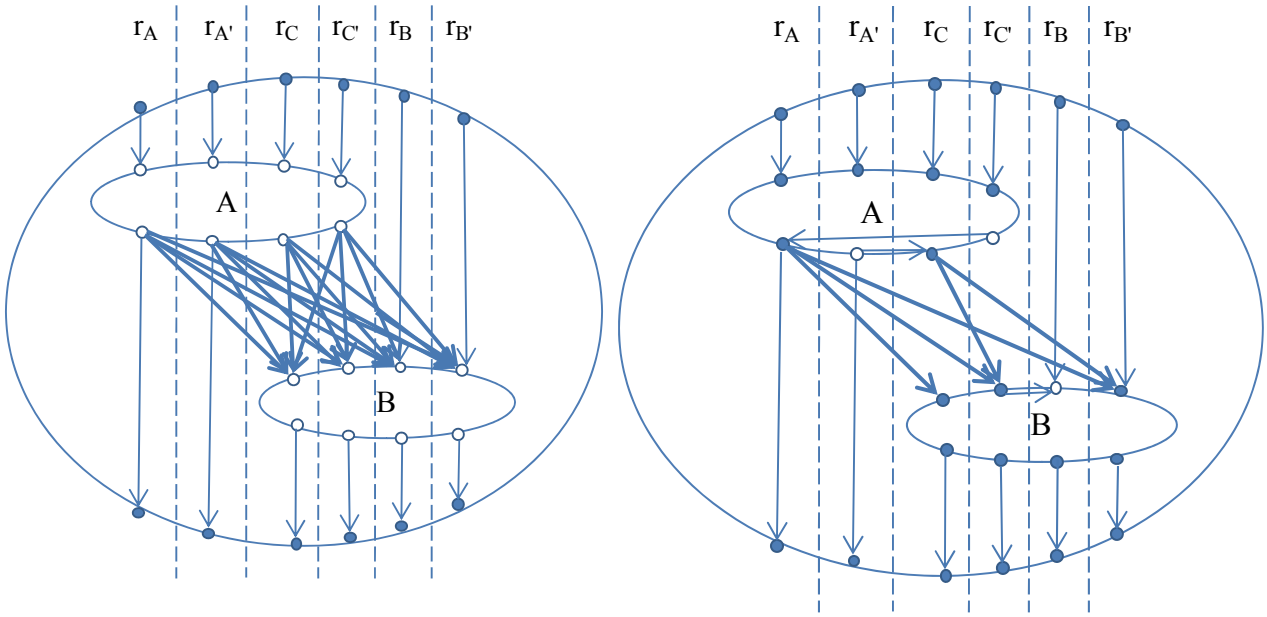
To realize the ordering relationships among different sub-collaborations in a distributed environment, one can use the flow messages and choice indication messages as described by Bochmann in [9]. In the formulas given in Chapter 5, we had assumed such messages would have a negligible delay and hence have not been included in our analysis. However, in reality, if we assume that these messages exist and have an appreciable delay, then we should modify our formulas and include in them the delay of these coordination messages. In the following, we assume that the delays of coordination messages are known (either measured or otherwise defined). They then can be taken into account by simply adding them (or convoluting them for stochastic delays) to the existing nominal execution time delays.

In this section, we discuss the impact of message transmission delays on the performance analysis of Chapter 5. We consider the cases of strict sequencing and alternatives. In the case of weak sequencing, there are no coordination messages, and in the case of weak and strict loops, there will be choice indication messages, but as the analysis is very similar to that of the alternative, we do not discuss these cases here.

7.3.1 Strict Sequence

For two collaborations, A and B, strictly sequenced, all actions must complete their executions before any action in collaboration B may begin. We modeled this by a synchronization event *S*, where event *S* is dependent on the ending events of all the roles of A, and the starting events of all roles of B are dependent on *S*. Using flow messages, this could be implemented as shown in Figure 24a, where there is a flow message from the ending event of every role in A to the starting event of every role in B.

Bochmann and Gotzhein in [10] pointed out that not all these dependencies need to be realized as messages. It is sufficient to have coordination messages from the terminating roles of A to all of the initiating roles of B, as shown in Figure 24b. Hence, flow messages are modeled from the terminating events in A, belonging to roles r_A and r_C to the initiating events in B, belonging to roles r_C, r_B and $r_{B'}$ and $r_{C'}$ and $r_{B'}$, respectively(note: there is no flow message from the terminating event belonging to r_C in A to the initiating event belonging to role r_C in B, as both the events belong to the same role).



a) Flow Messages defined in our CBD b) Flow messages defined in Bochmann [9]

Figure 24 – Message Types

If we assume Bochmann's method for specifying messages for strict sequencing as shown in Figure 24b, to include the message delays, we need to only revise Equation (15f) for case (1) of Table 2 as follows (note: a similar revision could also be applied to the stochastic formula):

$$\max_{x \in T(A)}\left({}^{(cp)}{}_{A}\Delta^{w}{}_{x} \quad + \quad \max_{y \in I'(B)}\left({}^{(cp)}m^{x}{}_{y} + {}^{(cp)}{}_{(mzz)B}\Delta^{y}{}_{z}\right)\right)$$

where T(A) is the set of terminating events of A, I'(B) is the set of initiating events of B, and $m^x{}_y$ is the message delay from event x to y and is equal to zero if x = y. The above revised formula is very similar to Equation (15f), except now we take the maximum over all the terminating events instead of all the ending events of A. This makes sense, as the terminating

event by definition takes longer than non-terminating ending events. Messages get passed from the terminating roles of A to all the initiating roles of B, which we account for by adding message delay m^x_y.

We note that taking the maximums over the terminating and the initiating roles only, and not over all ending and starting roles, as defined in Chapter 5 for Equation (15f) simplifies the calculations in general, since the number of terminating and initiating roles is often much smaller than the number of ending and starting roles.

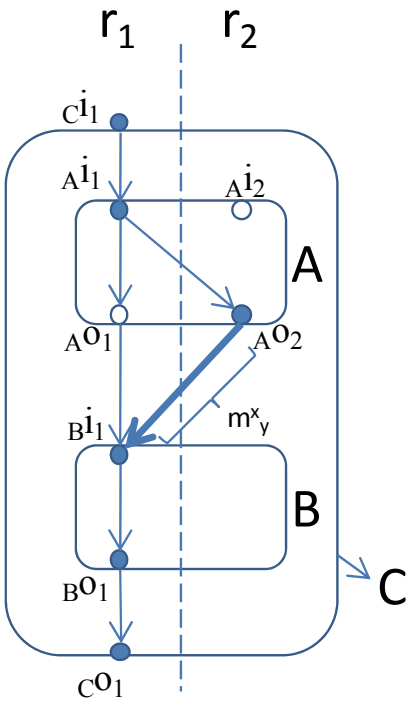


Figure 24c – Message Delays

For example, in Figure 24c, the fixed delay from the starting event ${}_{C}i_1$ to the ending event ${}_{c}o_1$ of role 1 in collaboration C, ${}_{C}\Delta^1_1$, can be calculated as (note: there is no delay from the ending event ${}_{A}o_1$ to ${}_{B}i_1$, as these two events both belong to the same role r_1):

$$ {}_{C}\Delta^1_1 = \max\left({}_{A}\Delta^1_1 + {}_{B}\Delta^1_1,\ ({}_{A}\Delta^1_2 + m^x_y + {}_{B}\Delta^1_1)\right), $$

where m^x_y is the message delay from event ${}_{A}o_2$ to ${}_{B}i_1$

7.3.2 Alternatives

For collaborations sequenced with the alternative operator, Bochmann [9] introduces the choice indication message to propagate the choice to a component that does not participate in the selected alternative. For example, Figure 25 shows collaborations A [] B, where role c* makes

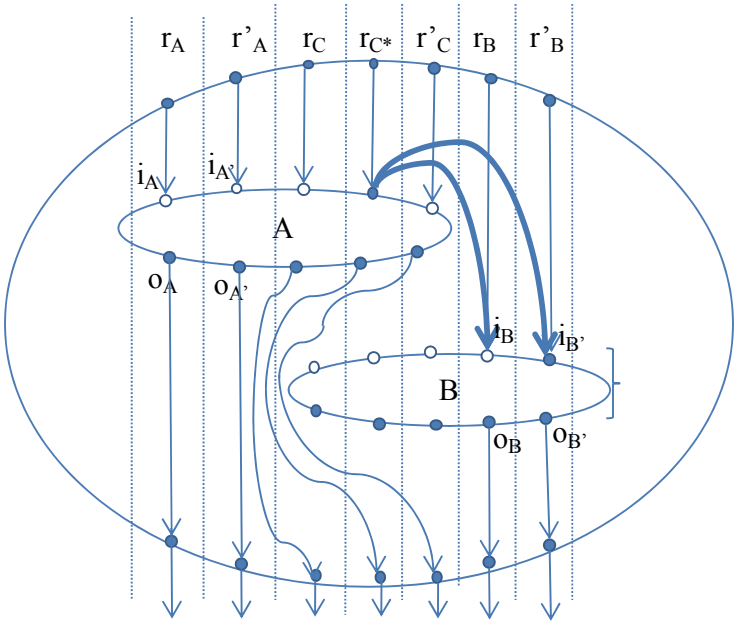


Figure 25 - Choice Indication Messages

the choice. If we assume that A is selected, then a choice indication message must be sent to the all the roles involved in B which are not involved in A.

If we consider Figure 25, and if sub-collaboration A was executed, then our earlier work had assumed the message delay from i_{C*} to o_B and to $o_{B'}$ to be zero. If we consider message delays to be non-zero, then the NETD from i_{C*} to o_B and to $o_{B'}$ provided A executes is $m^{c*}o_B$ $m^{c*}o_{B'}$ respectively, where $m^{c*}o_B$ and $m^{c*}o_{B'}$ is the message delay from i_{c*} to o_B and to $o_{B'}$, respectively.

In our analysis of Chapter 5, the value zero of Equations 32 for fixed delays of a single control flow path for the Alternative operator (Table 5) should be replaced by $m^w{}_z$ (the message delay from role w to role z).

7.4 Summary

We recognize we had to make certain assumptions to analyze the performance of composite collaborations in the earlier chapters. In this chapter, we discuss some of their implications and try to estimate the amount of error that may be caused by these assumptions. Assumptions are made for multiple simultaneous users of a single shared resource, for passive resources, for dependent probabilities and distributions, and for the existence of messages and impact of message delays on the formulas introduced in Chapter 5.

8. TOOL

We developed a tool called Performance Analysis Tool (PAT) to analyze the performance of a global collaboration based on the performance of the constituent sub-collaborations, using the formulas defined in Chapters 6 and 7. This tool, implemented in Eclipse (Java), accepts a UML Activity Diagram, with performance annotations, transforms it into a corresponding simplified data structure and analyzes the performance. This approach is described in Figure 26. The input detailing the UML Activity Diagram is created by the user using a graphical editor tool such as Rational Software Architect (RSA) [70]. This model is annotated with performance annotations using UML Constraints. The annotated model is then exported in an XMI format, which the Eclipse XMI reader reads in, and generates the internal data structure using the UML Meta Model Library. The appropriate information is extracted from this data structure and transformed into a Simplified Data Structure. The performance of this Simplified Data Structure is then analyzed, and the results are generated either as a console output or a file output (depending on the user's choice).

Any UML graphical editor can be used to generate the input for the transformation and performance analysis, if it uses the same XMI schema as Rational Software Architect's. However, if other schemas are used, then only the reading of the input would need to be revised. Alternatively, if all the tools use the same schema and a standard XMI is implemented, then no issues should arise using another UML graphical editor.

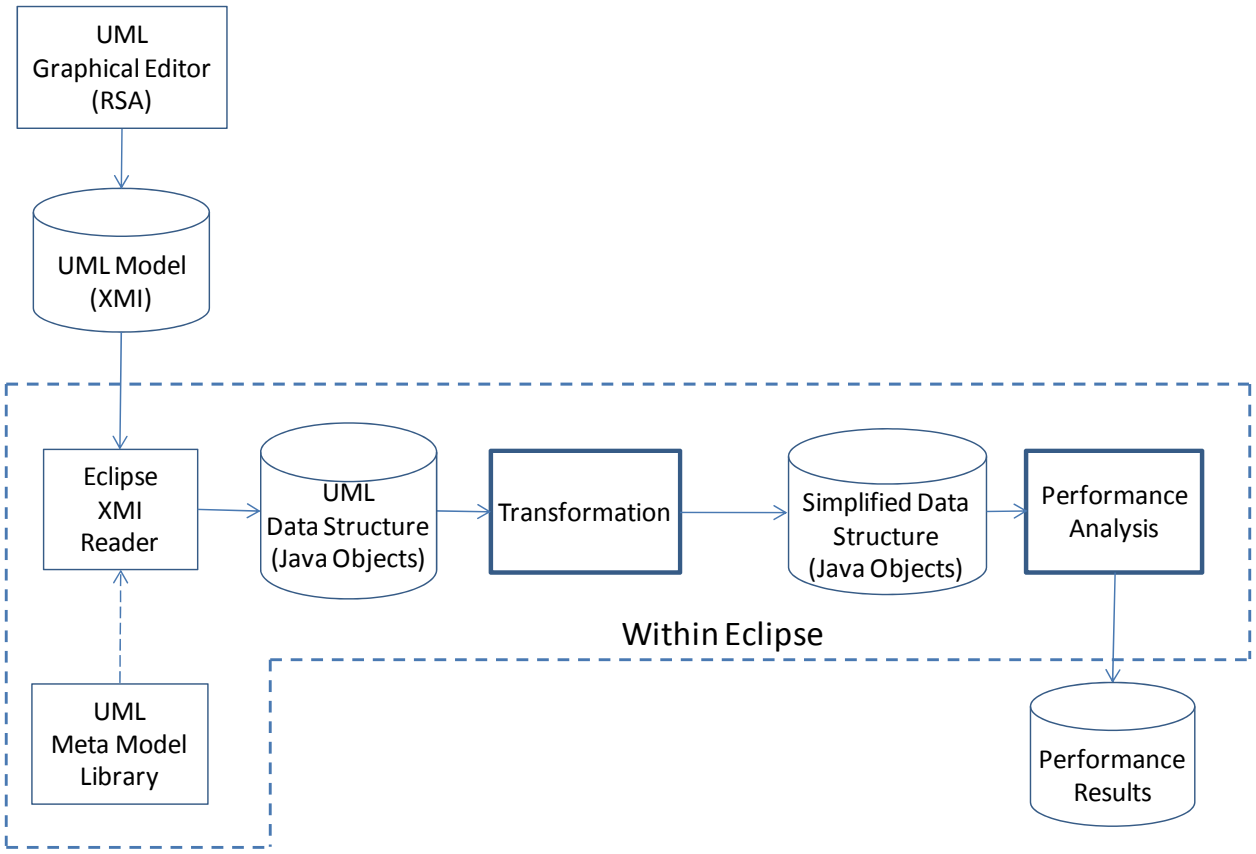


Figure 26 – Performance Analysis Tool

8.1 Software Development Environment

8.1.1 Rational Software Architect (RSA)

The IBM Rational Software Architect (RSA) is a design and development tool environment that allows software architects and senior developers to specify and maintain all aspects of the architecture of an application [70]. RSA leverages model-driven development with UML for creating well-architected applications and services. Built on top of the open-sourced Eclipse Platform, this tool is influenced by several open industry standards.

RSA is a complete design and a development toolset. This tool includes all the features of the IBM Rational Application Developer for WebSphere Software for building scalable Web services, Java, J2EE and other portable applications. This software has the capacity for the visualization and editing of J2EE, Java and C++ structures and behaviour through UML diagrams. It has support for UML 2.0 diagrams for analysis and design using Use Case, Class, Sequence, Activity, Composite Structure, State Machine, Communication, Component and Deployment Diagrams, which allows the user to capture and communicate all aspects of an application architecture using an industry standard. These diagrams can in turn be used for the generation of code from the design model. This transformation can be customized to tailor code generation patterns to meet the needs of an organization.

RSA can generate HTML, PDF and XML reports from UML designs. These documents are not only useful in the process of documentation but also for the interconnectivity with other tools. RSA can export user created UML 2.0 models in UML2 format, which in turn can be imported into the Eclipse Platform (with the UML2 plug-in) for further analysis and transformation. This powerful feature is extremely useful since this can produce the meta-objects of any given diagrams for further transformations.

8.1.2 Eclipse

IBM's Eclipse Platform is an extensible Integrated Development Environment (IDE) used to develop applications in Java. This open source tool is designed to facilitate tool

providers with mechanisms and rules to follow that would lead to seamlessly integrated tools. These mechanisms and rules are described by well-defined API interfaces, classes and methods.

Eclipse can provide support for the construction of a variety of tools for application development. By means of various plug-ins, this tool supports arbitrary content types such as Perl, Fortran, COBOL, C, C++, etc. With its ability to work with GUI and non-GUI based application development environments, it can run under a variety of operating systems, including Windows and Linux. Furthermore, it supports the integration of tools across different content types and tool providers.

8.1.2.1 Plug-ins

A plug-in, the smallest unit of the Eclipse Platform, can be developed and delivered separately. It is coded in Java and usually consists of Java code in a JAR library, some read only files, and other resources such as images, web templates, GUI components, etc. A plug-in does not necessarily need to contain code at all, such as the plug-ins which contributes to online help in the form of HTML pages. For interoperability between other plug-ins, each plug-in can declare any number of named extension points and any number of extensions to one or more extensions points in other plug-ins. The Platform Runtime, on start-up, determines the set of available plug-ins, leading it to build an in-memory plug-in registry. The extension declarations by name are matched with their corresponding extension point declarations, resulting in a plug-in registry which is available through the Platform API.

8.1.3 UML2 Plug-in

The UML2 project is an open-source project based on the Eclipse Modeling Framework from IBM. This is a useable implementation of the UML 2.0 meta-model to support the development of modeling tools. The UML-2 plug-in also provides test cases as a means of validating the specification, as well as to specify the validation rules for defining and enforcing levels of compliance. The plug-in provides a common XMI schema to assist the interchange of semantic models. This feature becomes extremely useful when importing models from various tools such as RSA. Not only can models be imported, but they can also be created programmatically in Eclipse. The UML2 plug-in provides application program interfaces (API) for the UML 2.0 meta-model it implements. This API can then be used to create user-defined models as well as to parse in already created models. This in-memory model can then be transformed according to the requirements of the users. Although this is all a Java-based implementation, the files are all stored in an XML format for easier interoperability.

8.2 Performance Analysis

In RSA, the dynamic behaviour of a Collaboration is modeled as an *extended* Activity Diagram with annotations specifying the various delays and sequence types. This is exported from RSA as an XMI file and our tool, the Performance Analysis Tool (PAT), parses this XMI file and renders it as Java Objects in Eclipse.

PAT calculates the performance of the global collaboration based on the options selected by user concerning the type of control flow path and the type of delays. The results from the calculation are directed either to the console output or to a file.

8.2.1 Input to the Tool (PAT)

8.2.1.1 Collaborations

As discussed in Section 2.3.2, the structure of Collaborations is defined in UML but UML has not defined a method to describe the behaviour of Collaborations. In Chapter 2, we explored different modeling paradigms to represent the dynamic behaviour of Collaborations and in Section 4.2, we proposed UML Activity Diagrams with partial orders to define the semantics of the behaviour of Collaborations. To model services and distributed workflows which may consist of *sub-services* and sequencing operators such as strict/weak sequences, alternatives, concurrency, strict/weak loops etc, we selected UML Activity Diagrams as the most suitable candidate. The semantic of activities is based on inherent token flow, where the token may contain some data, object or a locus of control. Executions are represented by action (or activity) nodes and dependencies are represented as edges. As collaborations may involve multiple components, UML Activities can span over multiple partitions, such that each partition represent a component.

We model collaborations as annotated UML Activity Diagrams in RSA. In our work, we analyze collaborations with multiple starting and ending points of execution which we called starting and ending events, respectively. UML Activity Diagrams can model these multiple

events – however, they restrict the timing of these events such that all the starting events within a given activity must occur at the same time and similar for the ending events. In our work, we do not impose such restrictions and would like to be able to model the multiple starting events when they occur at different times, and similar for ending events. Hence, we need to extend the semantics of UML Activity Diagrams to support these and other properties.

8.2.1.2 NETD

In each collaboration, we wish to model the Nominal Execution Time Delays between the starting and ending events of all the involved roles. To do this, first we model the starting and the ending event of a role as UML Input and Output pins. Each of these Input and Output pins is owned by a partition to which the starting and ending events belong. We use UML Constraints to specify the NETDs. These UML Constraints are associated with one Input pin and one Output pin. Therefore the NETD specified in the Constraint is the NETD between the Input pin (starting event) of the Partition (role) which owns this Input Pin and the Output pin (ending event) of the Partition (role) which owns this Output pin. The delays which can be specified are as follows:

- Fixed Delays: "fixed="
- Range of Delays: "max=" , "min="
- Stochastic Delays: "param1=", "param2=", "param3="

Specifying NETD as Fixed Delays and Range of Delays is quite self-explanatory. For Stochastic Delays, we assume the following:

i) NETDs in all the collaborations follow the same type of distribution. This is specified in a UML Constraint as:

"distributionType=<distribution type>"

where <distribution type> is a String.

ex. distributionType=*normal*

ii) Parameters param1, param2, and param3 are generic parameters, which can be used to specify the parameters of any distribution. For example, if the distribution type for an NETD is a normal distribution, then param1 can be *mean* and param2 can be *standard deviation*. However, if the distribution type is exponential, then param1 can be the *rate* and so on and so forth. For example, for a normal distribution, the mean and the standard deviation can be specified as param1 and param2, respectively, while no value is specified for param3:

ex. param1 = *0*

param2 = 0.5

For our tool, it is not incorrect to specify all (or some) of the above delay types, as long as the required delay type is specified. For example, if the tool is asked to calculate Range of Delays, it expects the NETDs specified as "max=" and "min=" and will ignore values for "fixed=" or "param1=" etc.

An example of a Collaboration with NETDs is shown in Figure 27.

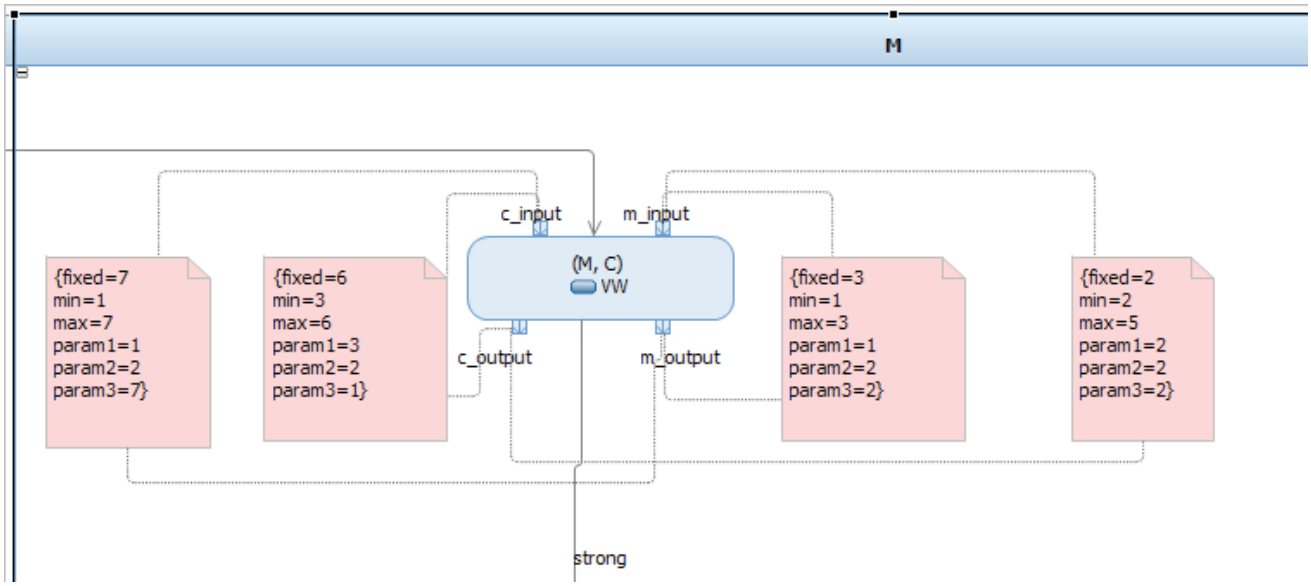


Figure 27 – Example of UML Constraints

8.2.1.3 Operators

One of the reason, we chose UML Activity Diagram to model the behaviour of collaborations is the similar set of operators UML Activity Diagrams offer and our requirement of operators for Collaborations. We can model these operators in RSA as follows.

8.2.1.4 Strict / Weak Sequences

As discussed, a sequence can be a strict sequence or a weak sequence. As there is no standard notation to differentiate between these two in UML Activities, we use the name attribute of the UML Control Flow in an Activity Diagram (also known as a *sequencing* operator) to define the type of a sequence, as shown in Figure 28. Hence the name of a control

flow can be either “strict” (also called “strong”) or “weak” – if the type is not specified, then strict sequence is assumed.

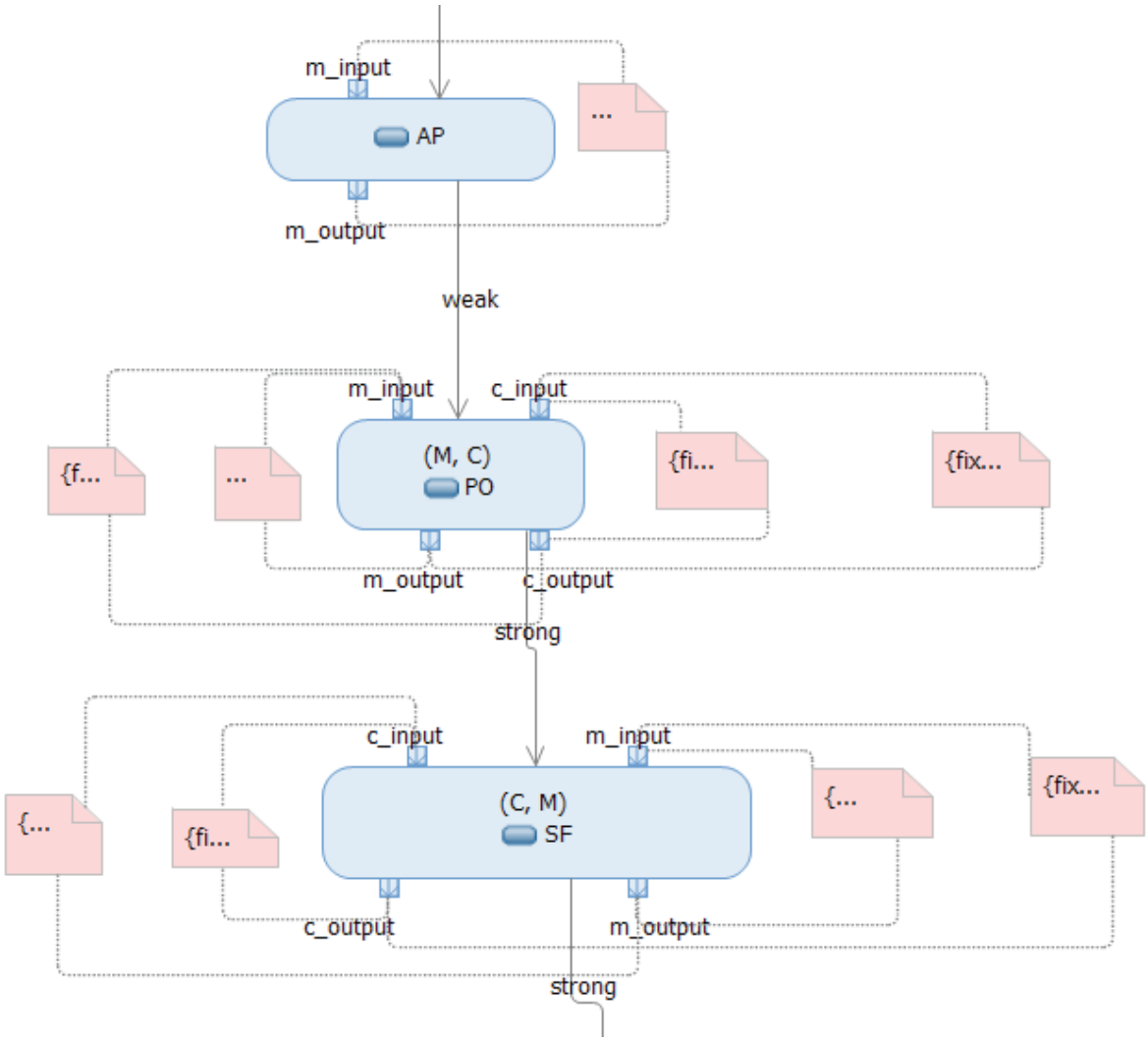


Figure 28 – Strict / Weak Sequence

8.2.1.5 Concurrency

Concurrency between two collaborations is modelled in the exact same manner as UML Activities are via, the Fork and Join Node as shown in Figure 29.

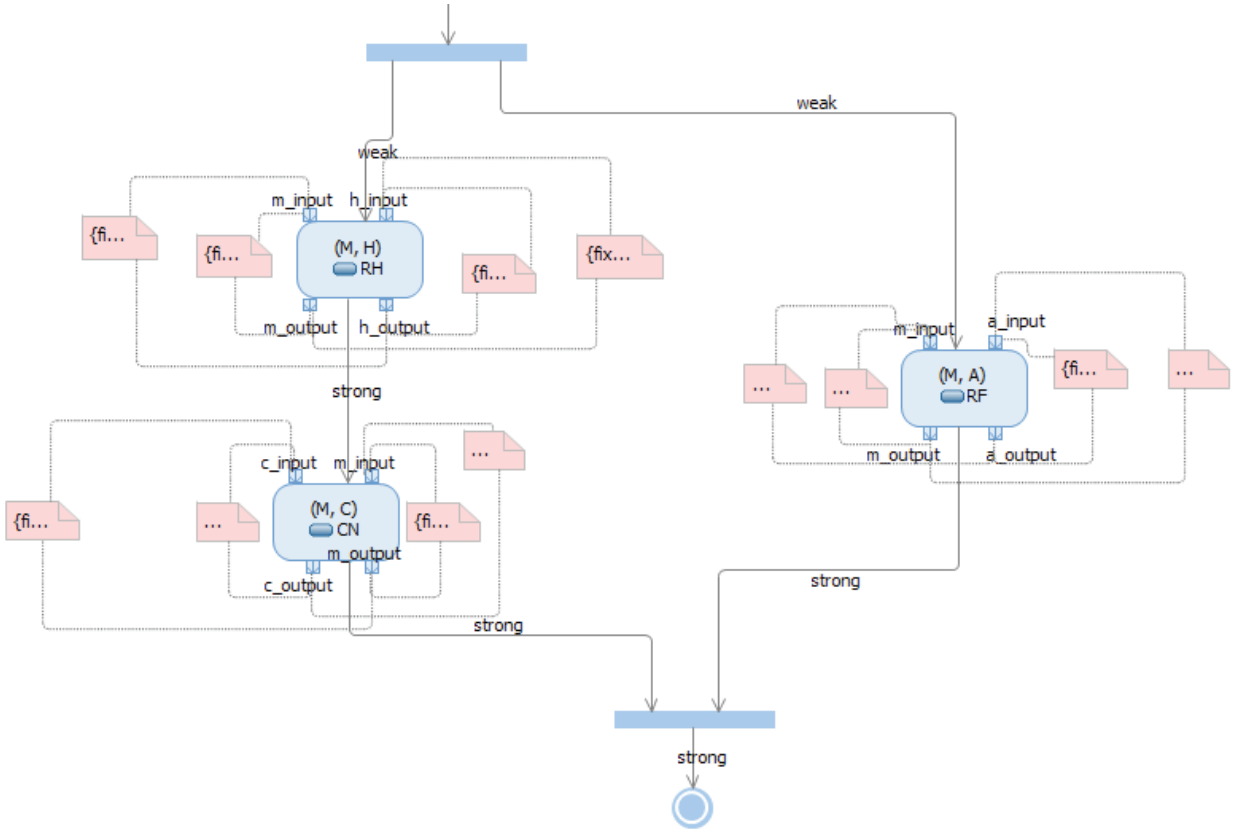


Figure 29 – Concurrency

8.2.1.6 Alternative

Alternatives are modeled using the UML Decision node. This decision node is owned by the partition that makes the choice between the multiple alternatives. If the control flow path

type and delay is selected as multiple control flow paths and stochastic delays, then probabilities need to be defined for each possible alternative. As shown in Figure 30, probabilities are specified as "prob=" in a guard condition on the control flow of each alternative path.

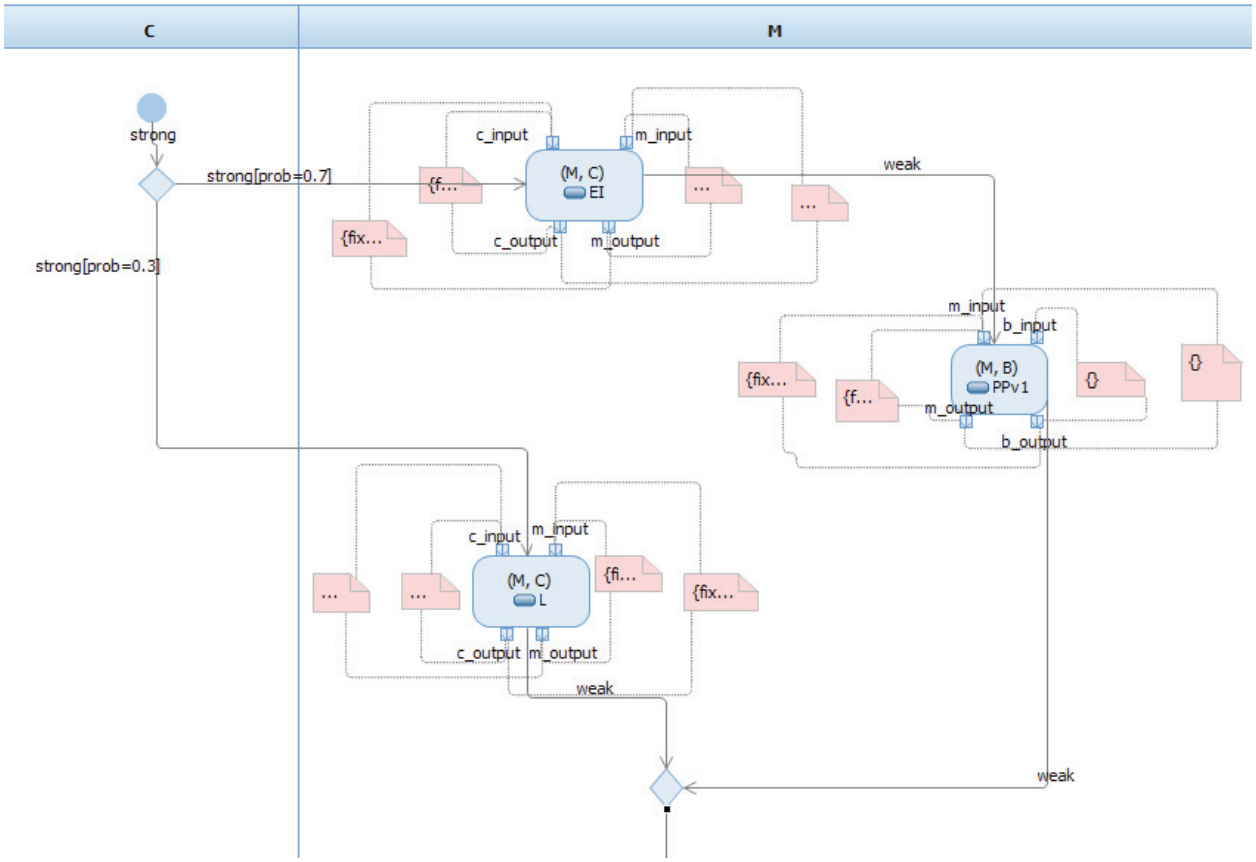


Figure 30 – Alternatives

8.2.1.7 Strict / Weak While Loops

Loops are modeled as a sequence of a UML Merge, followed by a UML Decision Node, as shown in Figure 31. Again, the Decision Node will be owned by the partition which makes the choice whether the loop continues or exits. For non-stochastic delays (fixed or range of

delays), we use the parameter "*n*" in the guard condition of the control flow to define the number of times the loops is to be executed. For stochastic delays, probabilities will exist for each path of the alternative. Even though in our model, it is not incorrect to specify both *n* and *probability* in the same model as done in Figure 31, only one of the two attributes will be used depending on the delay type that was selected by the user. The control flows after the DecisionNode are named to reflect the type of the loop – i.e. strict loop or weak loop.

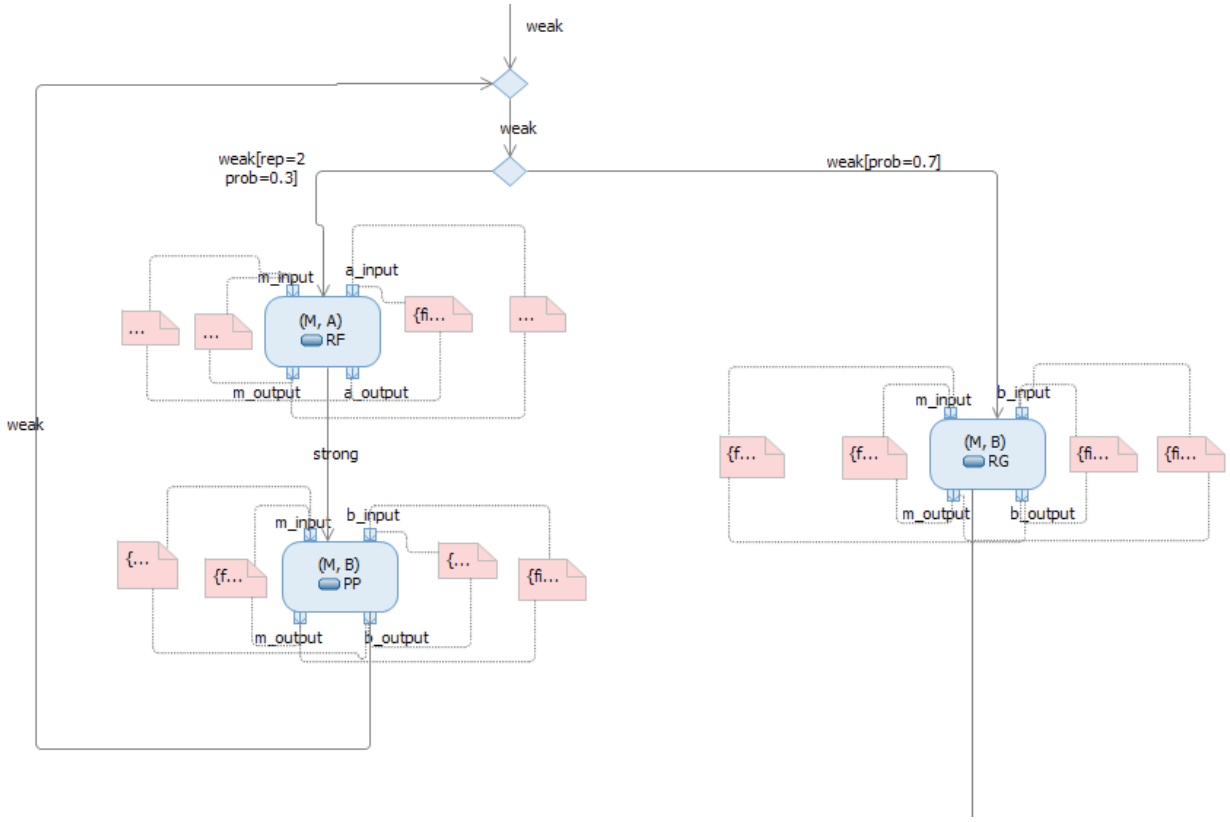


Figure 31 – Loops

8.2.2 Performance Analysis Tool (PAT)

After extracting the XMI model from RSA, we execute our Performance Analysis Tool. As shown in Figure 32, PAT offers the user to calculate the delay while considering either a single control flow path or multiple control flow paths.

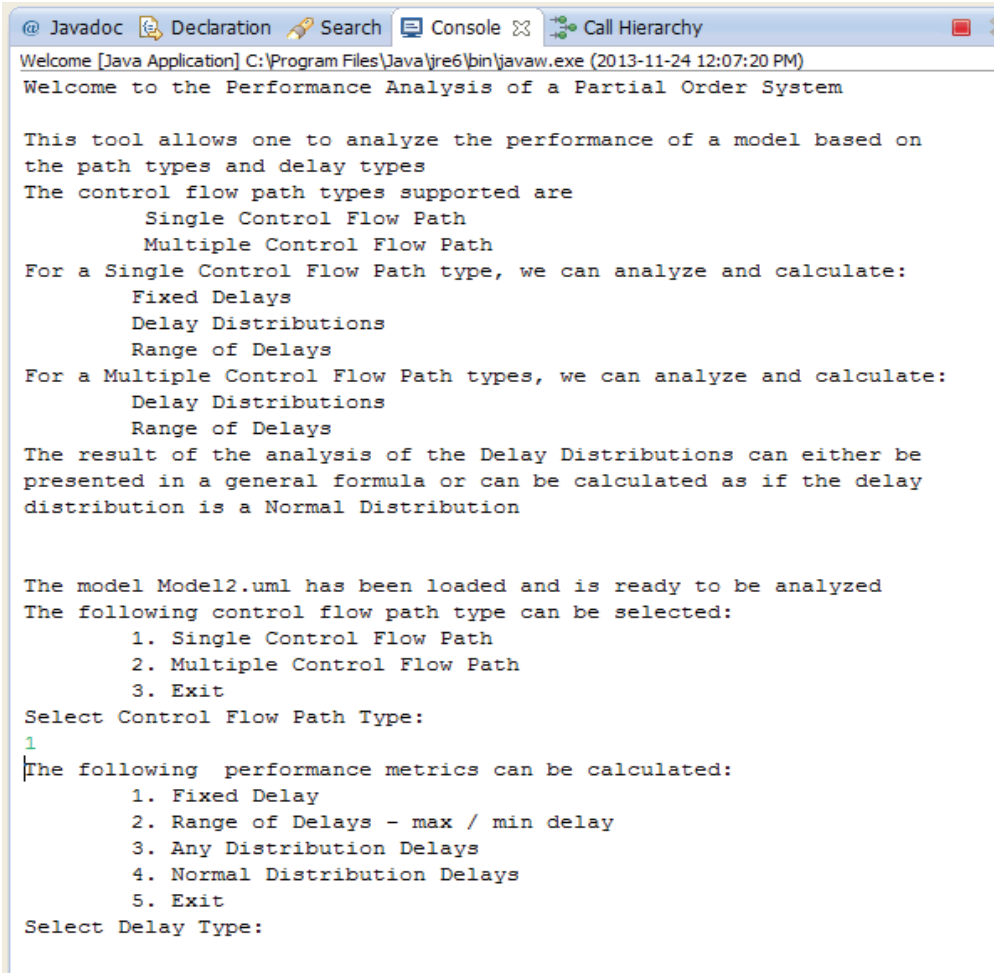


Figure 32 – Input Console

Similar steps are followed by the tool for both cases – the tool transforms the XMI file to a Simplified Data Structure and delays for the global collaboration are calculated based on the formulas from Chapter 5. However, when a single control flow path is selected, then the delay for a single path of execution can only be calculated and hence for any alternative paths of execution in the model, PAT preprocesses the XMI file and prompts the user to select the desired path of execution. For a single control flow path, PAT can calculate fixed, range and stochastic delays and for multiple control flow paths, PAT can calculate range and stochastic delays.

8.2.2.1 Simplified Data Structure (SDS)

The annotated UML meta-model is exported by RSA and is imported by the Eclipse XMI Reader. This model in the form of Java Objects is a quite complex model to work with. PAT begins by parsing this complex UML meta-model and transforming it into a Simplified Data Structure, a simple Java data structure consisting of activities with delay information and the sequencing.

We say that each activity is transformed into a type *PerfActivity* object. A *PerfActivity* is a class which, amongst other attributes, has the name of the activity which it represents and the delays between all the starting and ending events of the activity.

Similarly various sequence operators are represented by the following Java strings: *"STRICT_SEQUENCE", "WEAK_SEQUENCE", "CONCURRENCY", "ALTERNATIVE", "STRICT_LOOP"* and *"WEAK_LOOP"*.

The *Parser* creates a Java *LinkedList* to represent the global collaboration, in the following called *GC_LIST*. It parses the UML meta-model as follows:

i) For the Initial Node, it creates a *INITIAL* Java String

ii) For each activity it encounters, it creates a *PerfActivity* object and adds it to the *GC_LIST*.

iii) For each strict / weak sequencing operator it encounters, it creates the respective Java String and adds that String object to the list. Hence, the example in Figure 28 would result in SDS as shown in Table 10.

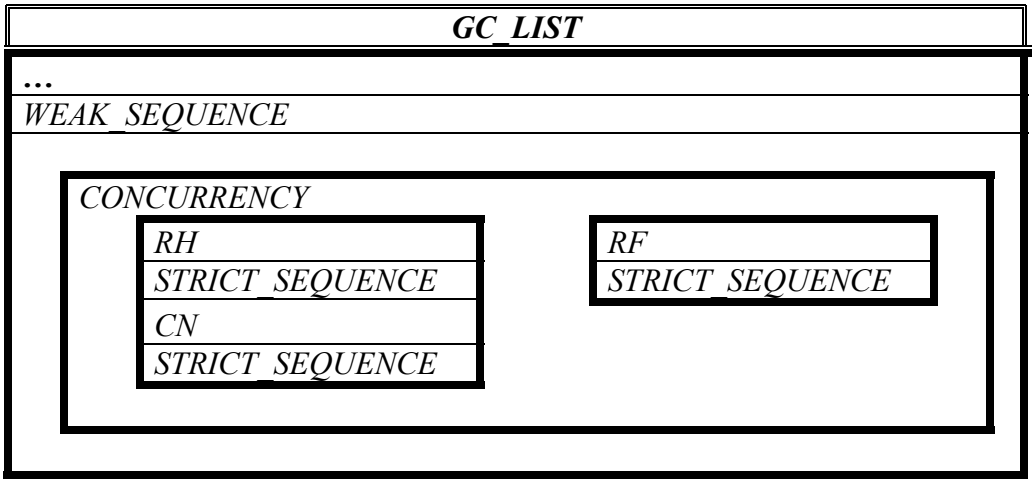
Table 10 – SDS of a Strict / Weak Sequence

GC_LIST	
AP	// AP is *Perf_Activity*
WEAK_SEQUENCE	// string object
PO	// PO is *Perf_Activity*
STRICT_SEQUENCE	// string object
SF	// SF is *Perf_Activity*

The elements inside the dark borders make up the contents of a single list (in this case GC_LIST).

iv) For the Concurrent operator, there are multiple paths of executions. Each possible path of execution is transformed into the Simplified Data Structure and added to a *sub-list*. Each *sub-list* is then added to the *GC_LIST*. So for example, the Simplified Data Structure (SDS) for Figure 29 would be as shown in Table 11.

Table 11 – SDS of Concurrency



The elements inside the dark borders make up the contents of a single list. In this case, there are 4 lists

list 1: RH, STRICT_SEQUENCE, CN, STRICT_SEQUENCE

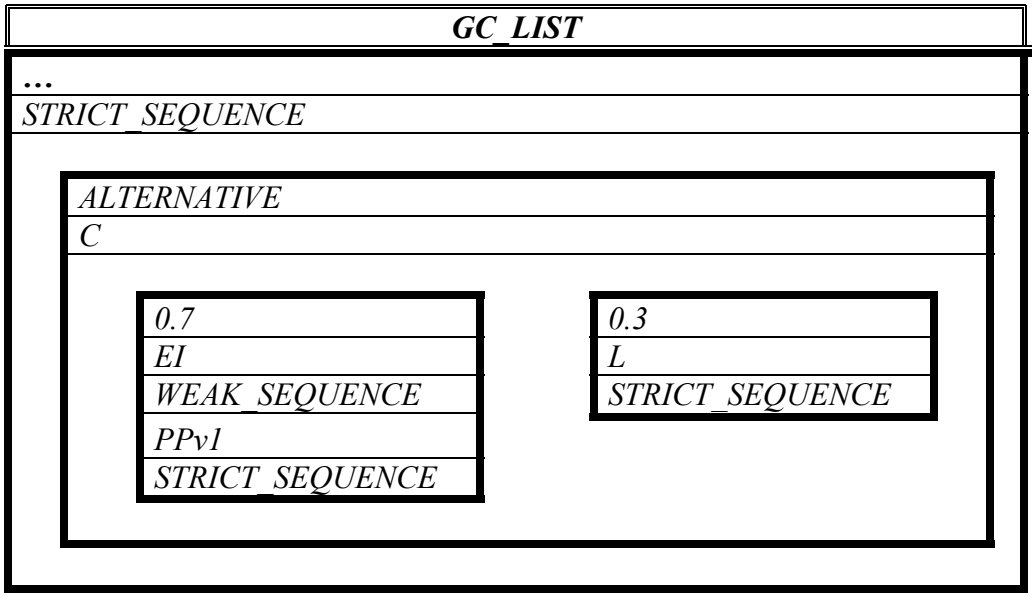
list 2: RF, STRICT_SEQUENCE

list 3: CONCURRENCY, <elements of list 1>, <elements of list 2>

list 4 (GC_LIST)**:** …, WEAK_SEQUENCE, <elements of list 3>

v) Alternative operator is transformed in the same way as the Concurrency operator with two additions. First, the role that makes the decision is identified and added to the Alternative list – this is a mandatory addition. Second, for stochastic delays, there are probabilities for the execution of each path. Hence the probability for each path is added to each path's *sub-list (EachPathList)* – this is optional as fixed and range of delays do not require probabilities. So, for example, the model in Figure 30 would be transformed as shown in Table 12.

Table 12 – SDS of Alternative



The elements inside the dark borders make up the contents of a single list. In this case, there are 4 lists

list 1: 0.7, EI, WEAK__SEQUENCE, PPv1, STRICT_SEQUENCE

list 2: 03, L, STRICT_SEQUENCE

list 3: ALTERNATIVE, C, <elements of list 1>, <elements of list 2>

list 4 (GC_LIST)**:** …, STRICT_SEQUENCE, <elements of list 3>

In the Alternative List (List 3), the second element "C" identifies the role C which makes the decision for the path of alternate executions.

In List 1 and List 2, 0.7 and 0.3 are the respective probabilities for these paths.

vi) The loop (Strict / Weak) operator is transformed in the same way as the Alternative operator. There is a decision that is made at the beginning of the loop to check whether the body of the loop is going to iterate again. The role that

makes this decision is placed in the *Loop sub-list* – this is a mandatory parameter. The number of times the body of the loop iterates can be specified for non-stochastic

Table 13 – SDS of Loop

GC_LIST
…
WEAK_SEQUENCE

WEAK_LOOP
C
2

0.3
RF
STRICT_SEQUENCE
PP
WEAK_SEQUENCE

0.7
RG
…

delays. This number of repetition is also added to the *Loop sub-list*. For stochastic delays, probabilities are defined for each path succeeding this decision. These probabilities are added to each path's *sub-list.* So, for example, the model in Figure 31 would be transformed as shown in Table 13.

The elements inside the dark borders make up the contents of a single list. In this case, there are four lists

list 1: 0.7, RG, …

list 2: 0.3, RF, STRICT__SEQUENCE, PP, WEAK_SEQUENCE

list 3: WEAK_LOOP, C , 2, <elements of list 1>, <elements of list 2>

list 4 (GC_LIST)**:** …, WEAK_SEQUENCE, <elements of list 3>

In the Loop List (List 3), the first element identifies the type of loop – weak or strict. The second element "C" identifies the role C which makes the decision whether the body of the loop will iterate again. The third element identifies the number of times the body of the loop (list 2) will repeat.

The first element in List 2 is the probability of repeating the loop. The second element onwards is the body of the loop.

List 1 contains the activities which follows the body of the loop.

8.2.3 Output Generated by the Tool

Once the performance analysis is complete, PAT returns the delays from the starting event of every role to the ending event of every role in the global collaboration, along with the execution paths traversed to get the delay.

For each delay, PAT first identifies the source and the destination role between which the delay is calculated. Then it presents the type of delay that is calculated and finally the value of the delay is given. Next is the execution path that was taken by PAT to calculate the delay, which includes the activity name, denoted as "act:", roles of starting and ending events, denoted by "s" and "d" respectively – *for source and destination*.

A **critical path** is a path with the maximum delay [23]. PAT can calculate the maximum delay along its execution path (*critical path*) by calculating the range of delays for multiple control flow paths.

As shown partially in Figure 33, for the first delay, it indicates that the delay from the starting event of role B to the ending event of role D is 18.0 seconds. The path of execution to calculate this delay is "in activity PPv1, starting event of role B to the ending event of role B, then in activity CH, starting event of role B to the ending event of role B…" and so on and so forth.

```
For Source: B to Destination: D
        fixed delay is: 18.0
        Path:
                act:PPv1 s:B d:B
                act:CH s:B d:B
                act:REF s:B d:M
                act:PP s:M d:B
                act:CH s:B d:B
                act:REF s:B d:D

For Source: D to Destination: D
        fixed delay is: 12.0
        Path:
                act:CH s:D d:D
                act:REF s:D d:M
                act:PP s:M d:B
                act:CH s:B d:B
                act:REF s:B d:D
```

Figure 33 – Console Output

For stochastic delays, the NETD for the global collaboration is not evaluated but rather a formula is generated with the specified parameters which can be modified and used in other tools such as MATLAB for evaluation. An example of the output produced for a stochastic delay is as follows:

```
((param1: 1.0, param2: 2.0, param3: 7.0)  CONVOLUTE
(param1: 4.0, param2: 1.0, param3: 1.0))
MULTIPLY
((param1: 3.0, param2: 2.0, param3: 1.0)  CONVOLUTE
(param1: 2.0, param2: 1.0, param3: 5.0))
```

8.3 Testing

We have employed manual testing for this tool. Several formulas must be exercised to calculate the delay of each sequencing operator mentioned in Chapter 5. These formulas are based on the participation of the roles of the starting and ending events. For example for a single control flow, for a strict sequence, there are 4 distinct formulas – one for Case 1, one for Case 2, one for Case 3 and one for Case 4, as mentioned in Table 2. Similarly, for multiple control flow paths, for the strict while loop, there are 2 formulas – one formula for if the role of starting event is a Choice role and another formula for the remaining roles. Each of these different cases should be tested.

Based on these conditions, we formed over 460 test cases based on the various situations of strict / weak sequence, concurrency, alternative and weak/strict loop. These test cases are detailed in Annex B, where each "X" represents a test case.

Our tool was tested against the above mentioned test cases and has passed each of the test cases.

8.4 Limitations and Future Work

This tool analyzes a given UML Activity Model and calculates the performance of the global activity. There are still some limitations, some inherent in UML while others are specific to the tool.

- Multiple Delays: The body of a loop (weak or strict) consists of activities. We specify only a single set of NETDs for these activities. However, each activity may have a different NETDs for each iteration. Ideally, there should be *n* delays defined for the *n* executions for all the activities in the body of a loop.
- Single Role for Choice: In an alternative or a loop, the decision could be made theoretically by a set of roles as explained in Chapter 6. However, in this tool, we only support a single role to make the decision. This should be extended to allow multiple roles to make the decision.
- Console Output: The output is currently presented in textual format, whether it is a file or a console output. Moving forward, it would be great if one could export this output back to RSA and be able to rebuild the collaboration with the new delays automatically. This would be extremely useful if the analyzed collaboration is reused as a sub-collaboration in another abstract model.

- Two Separate Programs: Currently, one needs to export the UML Activity Model from RSA and manually run the Performance Analysis Tool, hence requiring RSA and Eclipse. It would be great if this tool would be a *plugged-in* to RSA such that one makes the model with annotation and is able to analyze the performance directly within RSA.
- Normal Distribution: In Chapter 6, we have presented approximations of Normal Distribution to analyze performance. However, these approximations have not been implemented yet, but rather the tool calculates the stochastic delays as an equation to be evaluated.
- Stochastic Delay Parameters: Currently, we allow up to 3 parameters to define any distribution. In the future, we hope to make this dynamic and perhaps either ask the user about the kind of distribution being specified or have the number of parameters be dynamic such that a user can enter as many parameters as he wishes.

8.5 Summary

In this chapter we introduce Performance Analysis Tool, a tool developed to implement the formulas introduced in Chapter 5. We discuss the underlying technology and the development environment used to build the tool. Furthermore, we also discuss the Simplified Data Structure, an intermediate model used to assist in analyzing the performance of the global distributed system. Testcases used to test the tool were also discussed as well as the limitations of the tool.

9. A CASE STUDY

In this chapter, we illustrate the use of the formulas from Chapter 5 by applying them to a fictitious case study to calculate the global delay for each involved participant.

9.1 Travel Management System

A Travel Management System **(M)**, as shown in Figure 34, is an online service used by people to reserve flights and hotels. This is comparable to existing online portals such as Expedia©, Priceline©, Travelocity©, etc. The Travel Management System is composed of three servers – a front end server serving the customers, a Hotel Management Server **(H)**, and an Air Travel Management Server **(A)**. The Hotel Management Server interfaces with two external airline companies partnered with the Travel Management System – namely Excellent Hotel **(E)** and Premium Hotel **(P)**. Similarly, the Air Travel Management Server also deals with two airline companies -- Fast Airways **(F)** and Reliable Airways **(R)**. The Travel Management System also has a working relationship with a local Bank **(B)** for all its financial transactions.

9.2 Structural Aspects of the Travel Management System

As discussed in Chapter 4, we found UML 2 Collaboration diagrams a suitable notation to illustrate the structural aspects of services. Collaborations inherit from both structural classifiers and behavioural classifiers. As a structural classifier, a Collaboration identifies the roles, connectors and collaboration uses. Figure 34 shows a Collaboration diagram describing the structure of the Travel Management System service, which consists of ten roles and sixteen

sub-collaborations among the roles. Even though we chose not to do so, we could have modeled the system using collaboration-uses (instead of sub-collaborations), and then these collaboration-uses would refer to other sub-collaborations which would have been defined separately.

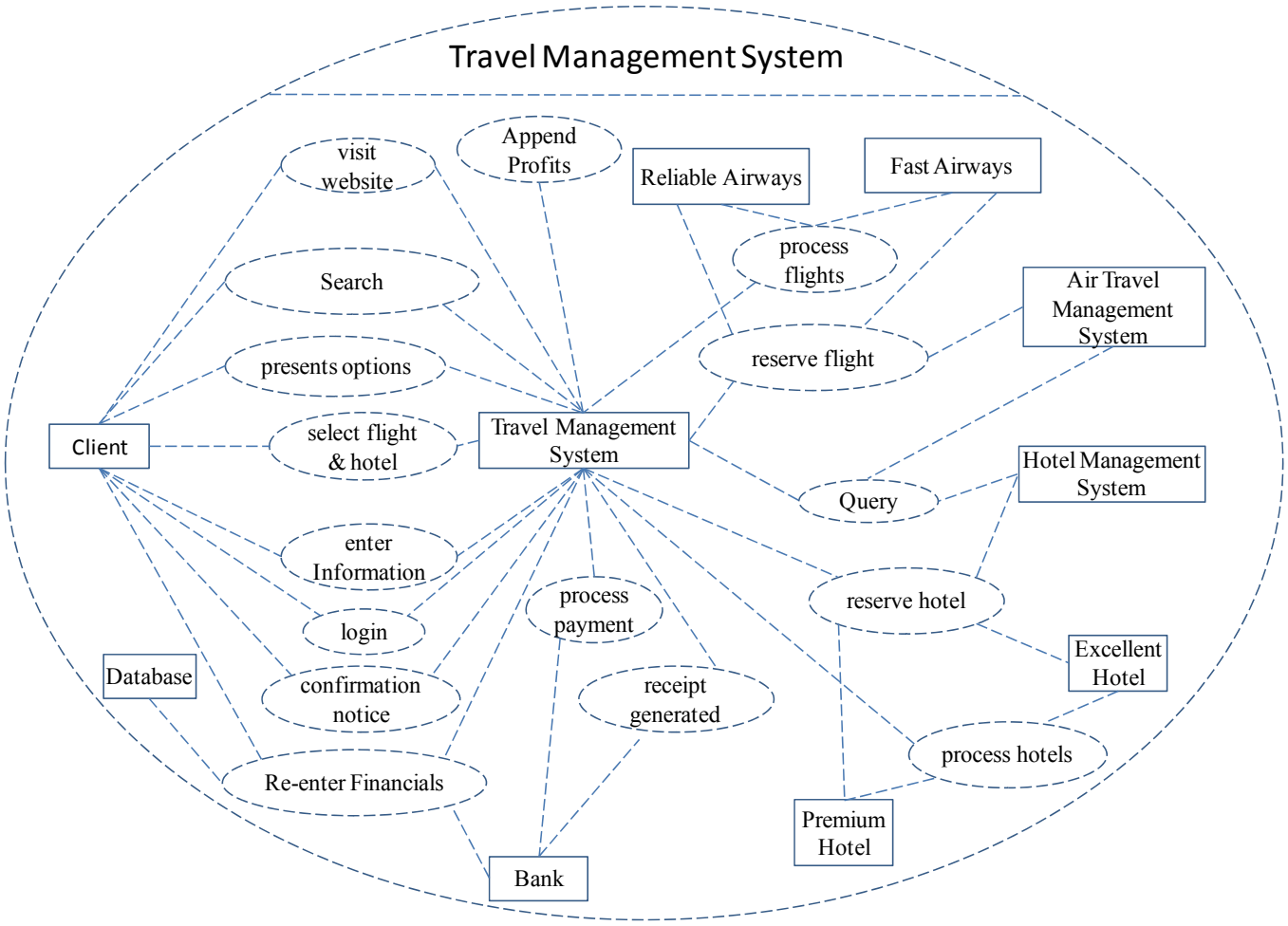


Figure 34 – Roles and Sub-Collaborations in Travel Management System

9.3 Behavioural Aspects of the Travel Management System

A typical scenario of a client making vacation plans using the Travel Management System is illustrated as a CBD in Figure 35. It shows the following order of sub-collaborations:

i) The client (***C***) visits the Travel Management System (***M***) website searching for a flight and a hotel. He/she enters the desired dates of travel and the destination. This is modeled by the *Visit Website* and *Search* sub-collaborations.

ii) ***M*** sends a request to the Air Travel Management System (***A***) and Hotel Management System (***H***) to search for the best suitable matches, modelled by the *Query* sub-collaboration.

iii) ***A*** contacts the two flight companies, Fast Airways (***F***) and Reliable Airways (***R***) with the request. These two companies respond back to ***A*** with corresponding price options, which get processed. All this is modelled by the *Processes Flight* sub-collaboration.

iv) *Processes Hotel* sub-collaboration models another similar request which is made by ***H*** to the two hotel companies Excellent Hotel (***E***), and Premium Hotel (***P***), which respond back with their availabilities and prices.

Note: Since the sub-collaborations *Processes Flight* and *Processes Hotel* are weakly sequenced, both of these sub-collaborations may execute in parallel.

Note: There is a strict sequencing after *Processes Hotel* and *Append Profits*. This means that all the roles involved in the previous collaborations need to complete their executions before *Append Profits* may execute. This shows the transfer of flight and hotel information from roles A and H respectively to M.

v) ***M*** processes the flight and the hotel information and adds its own profit to the price and presents it to the customer, shown by a weak sequence of the sub-collaborations *Append Profits* and *Present Options.*

vi) The client may find the presented choices unacceptable. This will cause the system to go back to Step ii) where the above is repeated with perhaps a revised set of dates and destinations.

vii) However, if the options are acceptable to the client, then the client can select the flight and the hotel, as modelled by sub-collaboration *Select Flight & Hotel*.

viii) The client may choose not to create a new account, and enter all information as a "guest" modelled by sub-collaboration *Enter Info* or he/she may login with an existing account, which would have all the customer information, including credit card information, which is modelled by sub-collaboration *Login*.

ix) ***M*** sends this information to the Bank (***B***) for payment to be processed, modelled by *Process Payment.*

x) If transaction is declined, the Bank notifies ***M***, which in turn notifies the customer. The customer enters a different credit card for authorization. Also, this new credit card information is updated in ***M***'s database (***dB***) simultaneously. This is modelled by sub-collaboration *Re-enter Financials.* This process keeps on repeating until the credit card is successfully authorized. Because of the weak sequencing, it is possible for the update to the database to take some time and lag behind. Hence, a scenario may exist where the database is being updated for the credit card number from the 2nd attempt, while the client may be entering credit card information for the 5th time.

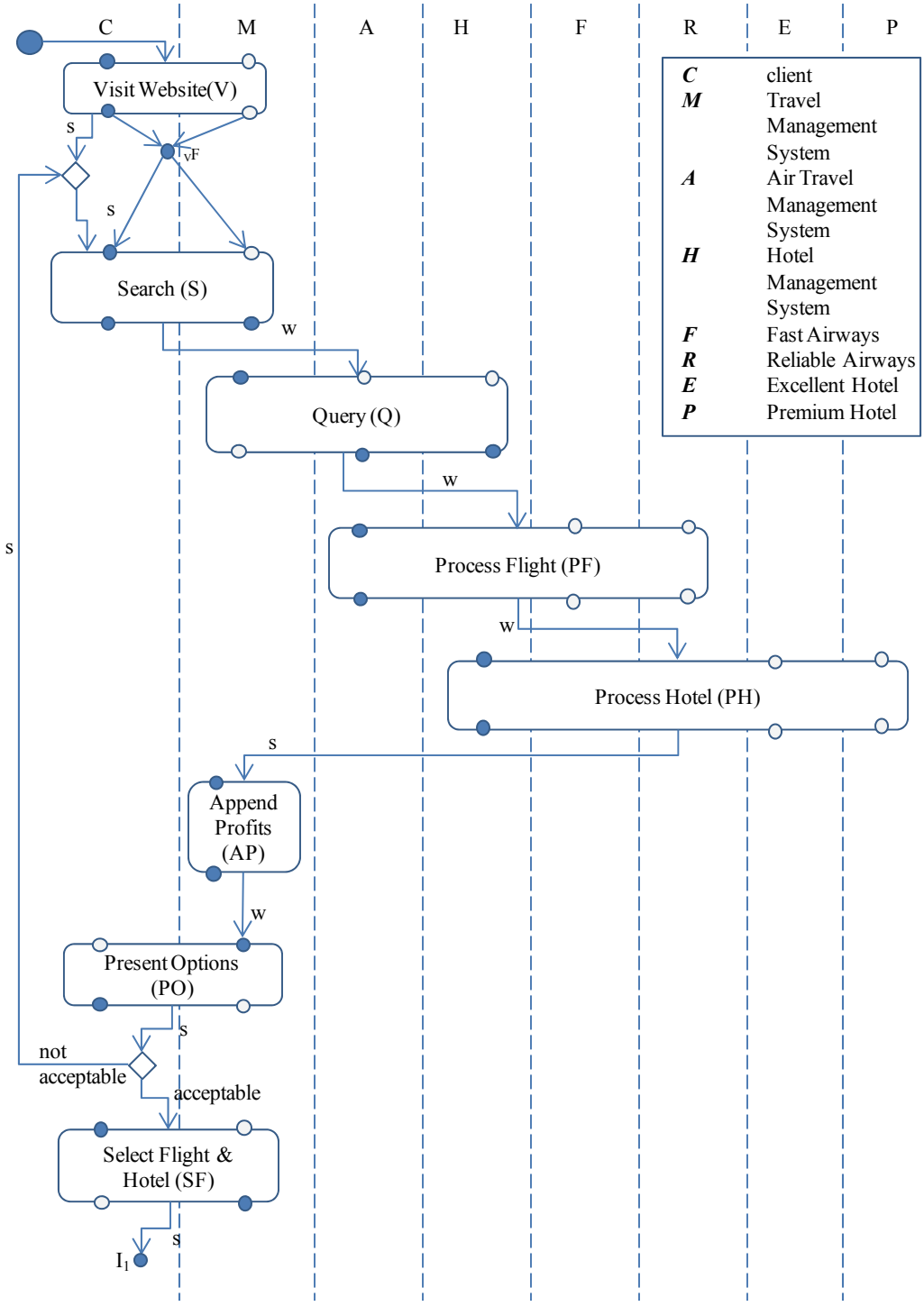


Figure 35a – Behavioural View of Travel Management System Collaboration (Part 1)

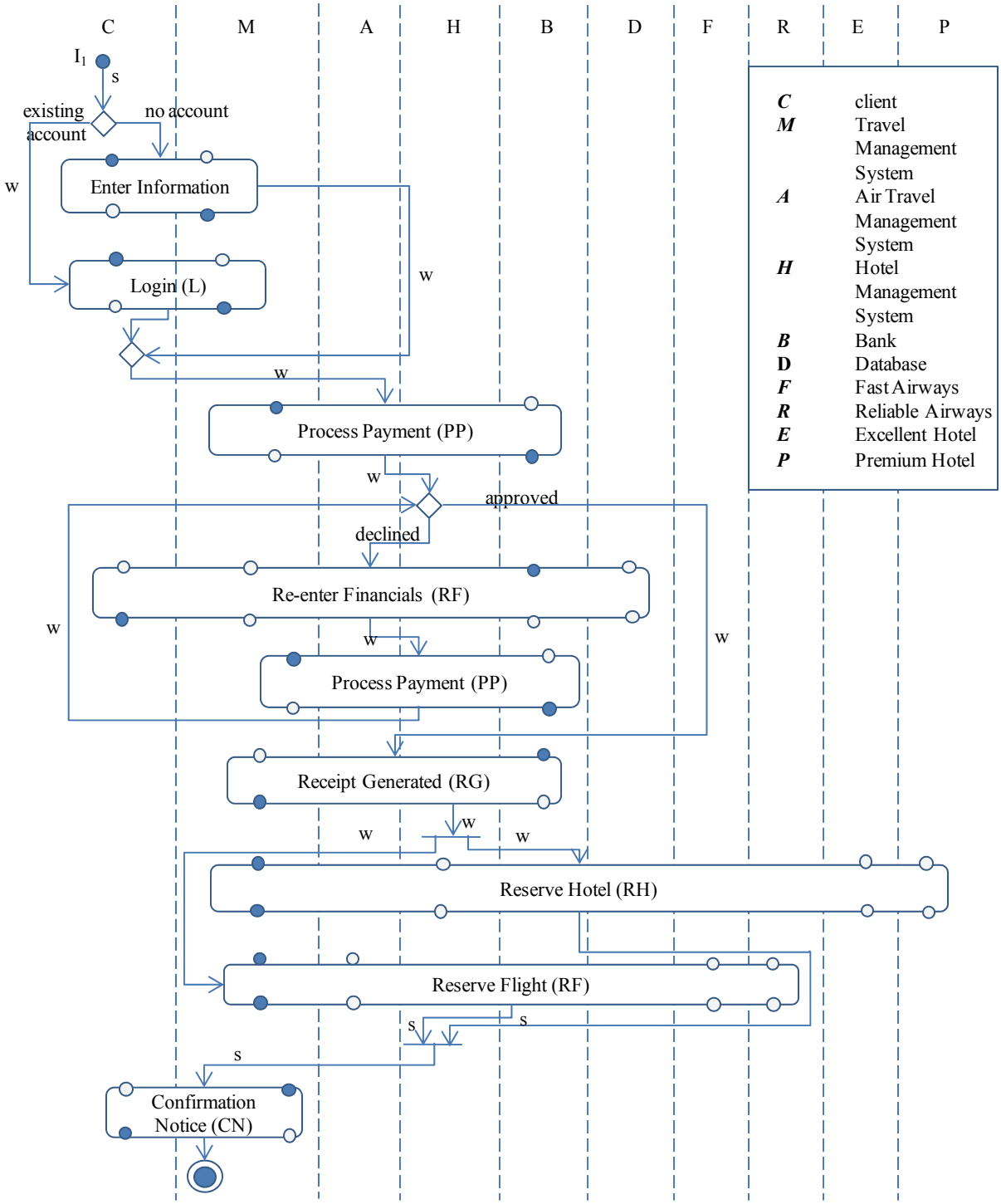


Figure 35b – Behavioural View (part 2)

xi) Once the transaction is approved, the Bank notifies ***M***. ***M*** concurrently reserves the flight modelled by sub-collaboration *Reserve Flight,* and the hotel as modelled by sub-collaboration *Reserve Hotel* with ***A*** and ***H***, respectively.

xii) A confirmation email is sent to the client, modeled by *Confirmation Notice*.

9.4 Performance Analysis

We aim to calculate the delay of the global collaboration, Travel Management System, based on the NETDs of the constituent sub-collaborations such as Visit Website, Query etc.

We analyze the initial three sub-collaborations, "Visit Website" (*V*), "Search" (*S*) and "Query" (*Q*) in this chapter. This analysis is based on the semantics and the formulas defined in Chapter 5. Results from the analysis of the remaining sub-collaborations can be found in Annex A.

9.4.1 Local Delays

The Nominal Execution Time Delay of the elementary sub-collaborations are called local NETDs (or local delays), annotated as ${}_{Z}\Delta^{x}_{y}$, where x is the role of the starting event, y is the role of the ending event and *Z* refers to the sub-collaboration to which these two roles belong (elementary sub-collaborations are sub-collaborations that cannot be decomposed into further sub-collaborations). Each of the sub-collaborations in Figure 35a-b is an elementary sub-collaboration. We assume that the local delays of all the sub-collaborations are known, either measured or specified.

9.4.1.1 Performance of the "Visit Website" (V) collaboration

There are two roles involved in the Visit Website collaboration, namely Client and Travel Management System. Each role has a starting event and an ending event. NETDs exist between the starting event of each role and the ending event of each role. This is summarized in Table 13. As these are local delays, we assume these delay are known, either specified or measured using the testing methodology from Section 4.5.

As Visit Website is the initial collaboration, we assume that there is no synchronization amongst the involved roles before the starting of the execution and the roles may start executing their actions as they become available. In a sense, we assume that there is weak sequencing prior to V. So a very typical scenario may have M being temporarily occupied and the role C would have to wait for M to become available.

Table 13 – Local Delays of sub-collaboration "Visit Website (V)"

Starting	Ending	**Local Delay of V**
C	C	${}_{V}\Delta^{C}{}_{C}$
C	M	${}_{V}\Delta^{C}{}_{M}$
M	M	${}_{V}\Delta^{M}{}_{M}$
M	C	${}_{V}\Delta^{M}{}_{C}$

9.4.2 Global Delays

In contrast to local delays, we say that a global collaboration is a composite collaboration that can be decomposed into sub-collaborations sequenced with other sub-collaborations. We call the NETDs of a global collaboration, global NETD (or global delays). We assume the global delay is not given but rather is calculated using the formulas from Chapter 5.

9.4.2.1 Performance of the collaborations "Visit Website" (V) followed by collaboration "Search"(S)

There is strict sequencing between *V* and *S* – the composite collaboration abstracting this strict sequence is denoted as "*VS*". There are two roles involved in this strict sequencing – *C* and *M*, both of which are common roles (i.e. both roles are participating in both collaboration V and S).

Strict sequencing, as defined earlier, specifies that all the actions in *V* must complete before any action in S may execute. Completion of all actions in V is specified by the event ${}_{V}F$. The delay from the starting event of role C of *V* to ${}_{V}F$ is the maximum of the delay from the starting event of C to the ending event of C and the starting event of C to the ending event of M, resulting in $\max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M})$.

We can calculate the delay from ${}_{V}F$ to the ending event of role C in S, by taking the maximum of the delays from the starting event of C to the ending event of C in S, and the starting event of M to the ending event of C in S, i.e. $\max({}_{S}\Delta^{C}{}_{C}, {}_{S}\Delta^{M}{}_{C})$.

With this, we can calculate the delay from the starting event of C to the ending event of C for the composite collaboration comprised of sub-collaboration V strictly sequenced with sub-collaboration S. This is done by summing up the delay from starting event of C in S to vF, and the delay from vF to the ending event of C in S, i.e. $\max({}_V\Delta^C_C , {}_V\Delta^C_M) + \max({}_S\Delta^C_C, {}_S\Delta^M_C)$ giving the total delay from the starting event of role C in *V* to the ending event of role C in *S*.

Similar calculations can be done to calculate the remaining three delays ${}_{VS}\Delta^C_M$, ${}_{VS}\Delta^M_M$, ${}_{VS}\Delta^M_C$, where *VS* is the composite collaboration abstracting the strict sequencing between *V* and *S*, as shown in Table 14.

Table 14 – Delays of Visit Website (V) Strictly Sequenced with Search(S)

Starting Role	**Ending Role**	**Global Delay**	
		Deterministic	**Stochastic**
C	C	$\max({}_V\Delta^C_C , {}_V\Delta^C_M) + \max({}_S\Delta^C_C, {}_S\Delta^M_C)$	$({}_V\Delta^C_C(t) * {}_V\Delta^C_M(t)) \otimes ({}_S\Delta^C_C(t) * {}_S\Delta^M_C(t))$
C	M	$\max({}_V\Delta^C_C , {}_V\Delta^C_M) + \max({}_S\Delta^C_M, {}_S\Delta^M_M)$	$({}_V\Delta^C_C(t) * {}_V\Delta^C_M(t)) \otimes ({}_S\Delta^C_M(t) * {}_S\Delta^M_M(t))$
M	M	$\max ({}_V\Delta^M_M, {}_V\Delta^M_C) + \max({}_S\Delta^M_M, {}_S\Delta^C_M)$	$({}_V\Delta^M_M(t) * {}_V\Delta^M_C(t)) \otimes ({}_S\Delta^M_M(t) * {}_S\Delta^C_M(t))$
M	C	$\max ({}_V\Delta^M_M, {}_V\Delta^M_C) + \max({}_S\Delta^M_C, {}_S\Delta^C_C)$	$({}_V\Delta^M_M(t) * {}_V\Delta^M_C(t)) \otimes ({}_S\Delta^M_C(t) * {}_S\Delta^C_C(t))$

9.4.2.2 Performance of composite collaborations "Visit Website" and "Search" (VS) followed by collaboration "Query" (Q)

The composite collaboration *VS* is subsequently weakly sequenced with sub-collaboration Query *(Q)*. This is abstracted by "VSQ". As mentioned in Section 2.3.1, strict and weak sequences are not associative. Therefore it is important to define the order in which such sequences are composed. We give priority to the composition of strict sequences over weak sequences. In this example, this corresponds to the "left-to-right" composition, which we have been following up to now.

There are four roles involved in the VSQ – namely *C, M, A* and *H*, where $C \in I^{NC}(VS)$, $M \in I^{C}$, and $\{A, H\} \in I^{NC}(Q)$. Even though ${}_{Q}\Delta^{C}{}_{C}$ is $-\infty$ as C is not involved in Q, but as can be seen in Table 15, ${}_{VSQ}\Delta^{C}{}_{C}$ is defined by Equation (23) as ${}_{VS}\Delta^{C}{}_{C} = \max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M}) + \max({}_{S}\Delta^{C}{}_{C}, {}_{S}\Delta^{M}{}_{C})$.

Again ${}_{Q}\Delta^{C}{}_{M}$ is $-\infty$ as C is not involved in Q, but due to the common role M (the only common role), there exists a path of dependency from C→M in VS and M→M in Q. Hence, according to Equation (23), the ${}_{VSQ}\Delta^{C}{}_{M} = {}_{VS}\Delta^{C}{}_{M} + {}_{Q}\Delta^{M}{}_{M} = \max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M}) + \max({}_{S}\Delta^{C}{}_{M}, {}_{S}\Delta^{M}{}_{M}) + {}_{Q}\Delta^{M}{}_{M}$. Other delays from the starting event of C are calculated in similar fashion.

As per Equation (23), delays for the abstract collaboration VSQ for roles from $I^{NC}(Q)$ to O(Q), are equal to the NETD measured in sub-collaboration *Q*, such as ${}_{VSQ}\Delta^{A}{}_{A} = {}_{Q}\Delta^{A}{}_{A}$, ${}_{VSQ}\Delta^{A}{}_{M} = {}_{Q}\Delta^{A}{}_{M}$, ${}_{VSQ}\Delta^{A}{}_{H} = {}_{Q}\Delta^{A}{}_{H}$, ${}_{VSQ}\Delta^{H}{}_{A} = {}_{Q}\Delta^{H}{}_{A}$, ${}_{VSQ}\Delta^{H}{}_{M} = {}_{Q}\Delta^{H}{}_{M}$, ${}_{VSQ}\Delta^{H}{}_{H} = {}_{Q}\Delta^{H}{}_{H}$.

Finally, delays from the starting events of the common role of VS to the ending events of all the roles in Q can be calculated using Equation (21), by taking the sum of ${}_{VS}\Delta^{M}_{M}$ and ${}_{Q}\Delta^{M}_{X}$, where $X \in O(Q)$. Hence ${}_{VSQ}\Delta^{M}_{M} = \max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{M}, {}_{S}\Delta^{C}_{M}) + {}_{Q}\Delta^{M}_{M}$, other delays can be calculated as: ${}_{VSQ}\Delta^{M}_{C} = \max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{C}, {}_{S}\Delta^{C}_{C})$, ${}_{VSQ}\Delta^{M}_{A} = \max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{M}, {}_{S}\Delta^{C}_{M}) + {}_{Q}\Delta^{M}_{A}$ and ${}_{VSQ}\Delta^{M}_{H} = \max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{M}, {}_{S}\Delta^{C}_{M}) + {}_{Q}\Delta^{M}_{H}$.

Table 15 – Delays of VS Weakly Sequenced with Query(Q)

Starting Role	**Ending Role**	**Global Delay**	
		Deterministic	**Stochastic**
M	M	$\max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{M}, {}_{S}\Delta^{C}_{M}) + {}_{Q}\Delta^{M}_{M}$	$({}_{V}\Delta^{M}_{M}(t) * {}_{V}\Delta^{M}_{C}(t)) \otimes ({}_{S}\Delta^{M}_{M}(t) * {}_{S}\Delta^{C}_{M}(t)) \otimes {}_{Q}\Delta^{M}_{M}(t)$
M	A	$\max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{M}, {}_{S}\Delta^{C}_{M}) + {}_{Q}\Delta^{M}_{A}$	$({}_{V}\Delta^{M}_{M}(t) * {}_{V}\Delta^{M}_{C}(t)) \otimes ({}_{S}\Delta^{M}_{M}(t) * {}_{S}\Delta^{C}_{M}(t)) \otimes {}_{Q}\Delta^{M}_{A}(t)$
M	H	$\max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{M}, {}_{S}\Delta^{C}_{M}) + {}_{Q}\Delta^{M}_{H}$	$({}_{V}\Delta^{M}_{M}(t) * {}_{V}\Delta^{M}_{C}(t)) \otimes ({}_{S}\Delta^{M}_{M}(t) * {}_{S}\Delta^{C}_{M}(t)) \otimes {}_{Q}\Delta^{M}_{H}(t)$
M	C	$\max({}_{V}\Delta^{M}_{M}, {}_{V}\Delta^{M}_{C}) + \max({}_{S}\Delta^{M}_{C}, {}_{S}\Delta^{C}_{C})$	$({}_{V}\Delta^{M}_{M}(t) * {}_{V}\Delta^{M}_{C}(t)) \otimes ({}_{S}\Delta^{M}_{C}(t) * {}_{S}\Delta^{C}_{C}(t))$
A	M	${}_{Q}\Delta^{A}_{M}$	${}_{Q}\Delta^{A}_{M}(t)$

A	A	${}_{Q}\Delta^{A}{}_{A}$	${}_{Q}\Delta^{A}{}_{A}(t)$
A	H	${}_{Q}\Delta^{A}{}_{H}$	${}_{Q}\Delta^{A}{}_{H}(t)$
A	C	$-\infty$	$-\infty$
H	M	${}_{Q}\Delta^{H}{}_{M}$	${}_{Q}\Delta^{H}{}_{M}(t)$
H	A	${}_{Q}\Delta^{H}{}_{A}$	${}_{Q}\Delta^{H}{}_{A}(t)$
H	H	${}_{Q}\Delta^{H}{}_{H}$	${}_{Q}\Delta^{H}{}_{H}(t)$
H	C	$-\infty$	$-\infty$
C	M	$\max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M}) +$ $\max({}_{S}\Delta^{C}{}_{M}, {}_{S}\Delta^{M}{}_{M}) + {}_{Q}\Delta^{M}{}_{M}$	$({}_{V}\Delta^{C}{}_{C}(t) * {}_{V}\Delta^{C}{}_{M}(t)) \otimes$ $({}_{S}\Delta^{C}{}_{M}(t) * {}_{S}\Delta^{M}{}_{M}(t)) \otimes {}_{Q}\Delta^{M}{}_{M}(t)$
C	A	$\max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M}) +$ $\max({}_{S}\Delta^{C}{}_{M}, {}_{S}\Delta^{M}{}_{M}) + {}_{Q}\Delta^{M}{}_{A}$	$({}_{V}\Delta^{C}{}_{C}(t) * {}_{V}\Delta^{C}{}_{M}(t)) \otimes$ $({}_{S}\Delta^{C}{}_{M}(t) * {}_{S}\Delta^{M}{}_{M}(t)) \otimes {}_{Q}\Delta^{M}{}_{A}(t)$
C	H	$\max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M}) +$ $\max({}_{S}\Delta^{C}{}_{M}, {}_{S}\Delta^{M}{}_{M}) + {}_{Q}\Delta^{M}{}_{H}$	$({}_{V}\Delta^{C}{}_{C}(t) * {}_{V}\Delta^{C}{}_{M}(t)) \otimes$ $({}_{S}\Delta^{C}{}_{M}(t) * {}_{S}\Delta^{M}{}_{M}(t)) \otimes {}_{Q}\Delta^{M}{}_{H}(t)$
C	C	$\max({}_{V}\Delta^{C}{}_{C}, {}_{V}\Delta^{C}{}_{M}) +$ $\max({}_{S}\Delta^{C}{}_{C}, {}_{S}\Delta^{M}{}_{C})$	$({}_{V}\Delta^{C}{}_{C}(t) * {}_{V}\Delta^{C}{}_{M}(t)) \otimes$ $({}_{S}\Delta^{C}{}_{C}(t) * {}_{S}\Delta^{M}{}_{C}(t))$

9.4.2.3 Performance Results of the sequence "Visit Website", "Search", and "Query"

Table 16 shows the local NETDs between the starting events of each role to the ending events of each role for all the roles involved in the three sub-collaborations, Visit Website, Search and Query. Most of these delays are self-explanatory, for example, in sub-collaboration Visit Website, delay from the starting event of role Client (C) to the ending event of the role Client (C) is 10 seconds, and so on. Some of these delays are $-\infty$, for example, from the starting event of role Client (C) to the ending event of the role Client (C), as per Equation (12) these roles are either not involved in that sub-collaboration or dependencies do not exist between those events.

Table 16 – Local Delays (*in seconds)*

	Delay
${}_{V}\Delta^{C}{}_{C}$	10
${}_{V}\Delta^{C}{}_{M}$	15
${}_{V}\Delta^{M}{}_{M}$	2
${}_{V}\Delta^{M}{}_{C}$	4
${}_{S}\Delta^{C}{}_{C}$	60
${}_{S}\Delta^{C}{}_{M}$	45
${}_{S}\Delta^{M}{}_{M}$	30
${}_{S}\Delta^{M}{}_{C}$	35

	Delay
${}_{Q}\Delta^{M}{}_{M}$	1
${}_{Q}\Delta^{M}{}_{A}$	4
${}_{Q}\Delta^{M}{}_{H}$	4
${}_{Q}\Delta^{M}{}_{C}$	$-\infty$
${}_{Q}\Delta^{A}{}_{M}$	4
${}_{Q}\Delta^{A}{}_{A}$	3
${}_{Q}\Delta^{A}{}_{H}$	$-\infty$
${}_{Q}\Delta^{A}{}_{C}$	$-\infty$

	Delay
${}_{Q}\Delta^{H}{}_{M}$	4
${}_{Q}\Delta^{H}{}_{A}$	$-\infty$
${}_{Q}\Delta^{H}{}_{H}$	3
${}_{Q}\Delta^{H}{}_{C}$	$-\infty$
${}_{Q}\Delta^{C}{}_{M}$	$-\infty$
${}_{Q}\Delta^{C}{}_{A}$	$-\infty$
${}_{Q}\Delta^{C}{}_{H}$	$-\infty$
${}_{Q}\Delta^{C}{}_{C}$	$-\infty$

Table 17 – Global Delays the sequence of sub-collaborations “Visit Website”, “Search” and “Query” *(in seconds)*

	Global Delay
${}_{VSQ}\Delta^{M}_{M}$	max (2, 4) + max(30,45) + 1 = 50
${}_{VSQ}\Delta^{M}_{A}$	max (2, 4) + max(30,45) + 4 = 53
${}_{VSQ}\Delta^{M}_{H}$	max (2, 4) + max(30,45) + 4 = 53
${}_{VSQ}\Delta^{M}_{C}$	max (2, 4) + max(35,60) = 64
${}_{VSQ}\Delta^{A}_{M}$	4
${}_{VSQ}\Delta^{A}_{A}$	3
${}_{VSQ}\Delta^{A}_{H}$	$-\infty$
${}_{VSQ}\Delta^{A}_{C}$	$-\infty$
${}_{VSQ}\Delta^{H}_{M}$	4
${}_{VSQ}\Delta^{H}_{A}$	$-\infty$
${}_{VSQ}\Delta^{H}_{H}$	3
${}_{VSQ}\Delta^{H}_{C}$	$-\infty$
${}_{VSQ}\Delta^{C}_{M}$	max(10 , 15) + max(45,30) + 1 = 61
${}_{VSQ}\Delta^{C}_{A}$	max(10 , 15) + max(45,30) + 4 = 64
${}_{VSQ}\Delta^{C}_{H}$	max(10 , 15) + max(45, 30) + 4 = 64
${}_{VSQ}\Delta^{C}_{C}$	max(10 , 15) + max(60, 35) = 75

9.4.3 Remaining Sub-Collaborations

Detailed performance analysis for the remaining sub-collaborations, based on the formulas from Chapter 5, can be found in Annex A. We abstract the behaviour of the Travel Management System of Figure 35a-b as collaboration D. Using our tool, Performance Analysis Tool, we calculated the fixed delays from the starting event to the ending event of each role. As the Travel Management System has an alternative path of execution, our tool calculates the delay for each path of execution and the results are presented in Tables 18 and 19 of Annex A.3.

9.4.4 Discussion

The performance analysis of the global "Travel Management System" can be used to identify a number of things:

- In many cases, "server time" is contracted from *server-provider* companies such as Amazon and IBM, where the contracts stipulate the use of the server where the charges are based on the resources consumed such as server–hours or amount of data–transfer [2]. If the amount of time typically required for a service is known, then this information can be used to better negotiate with the *service-providers.*

  For example, in the Travel Management System, our tool has calculated the delays from the starting events to the ending events for all the roles. If we assume that the time of the starting events for all the roles is the same, then we can determine the amount of time required for a server by taking the maximum of the NETDs: $\max_{i \in I(D)} ({}_{D}\Delta^{i}_{o})$, where D is the

global collaboration. For example, from Equation (3), the time when the ending event of role A (Air Travel Management) occurs is:

$$T_B = \max ({}_D\Delta^C_{A,\,D}\Delta^M_{A,\,D}\Delta^A_{A,\,D}\Delta^H_{A,\,D}\Delta^B_{A,\,D}\Delta^D_{A,\,D}\Delta^Q_{A,\,D}\Delta^R_{A,\,D}\Delta^E_{A,\,D}\Delta^S_A)$$

$$T_B = \max (\mathit{123.5}, \mathit{119.5}, \mathit{116.5}, \mathit{113.5}, \mathit{33.5}, 27.5, \mathit{106.5}, \mathit{110.5}, \mathit{104.5}, \mathit{102.5})$$

$$= 123.5$$

Therefore, if the time of the starting events is the same for all the roles, then the amount of time required for Air Travel Management to satisfy a single client is 123.5 seconds.

This can be determined for each of the involved roles.

- If the Air Travel Management sub-system is a third party sub-system and this service can be provided by multiple *service-providers,* then our tool can calculate the performance impact of these options on our global collaboration. In this way, the tool can analyze the effects on performance due to addition, removal and replacement of any software sub-systems within a given workflow.
- In many cases, the *end-to-end* delay typically describes the delay from the beginning to the ending of execution of actions in a behaviour involving only a single role. In this context with multiple roles, the *end-to-end* delay is the delay from the first starting event to the last ending event in the global collaboration. Obviously the first starting event depends on the availability of each role. If we assume that all the roles become available at the same time, i.e. time of the starting events for all the roles is the same, then the *end-to-end* delay is the maximum delay from the starting to the ending event for all the roles involved:

$$\textit{end-to-end delay} = \max_{M \,\varepsilon\, I(D),\, N \,\varepsilon\, O(D)} ({}_D\Delta^M_N)$$

The end-to-end delay obviously depends on the control flow path for fixed delays. For the control flow path where sub-collaboration EI was executed, the end-to-end delay is 130.5 secs, while when sub-collaboration L was executed, the end-to-end delay is 128 secs, both from the starting event of role C to the ending event of role M.

- In an execution of a workflow, a critical path is the longest necessary path of execution from the initial event to the last event, producing the above mentioned *end-to-end* delay. Any increase of the delay of an action along the critical path will also increase the *end-to-end* delay by the same amount.

  Similarly, the end-to-end delay can be reduced by reducing the delays of some actions on the critical path. This can be done by sub-collaborations with slack time, providing resources to sub-collaborations on the critical path in order to improve the overall process.

  As we do not model the allocation of resources in our modeling specification, we do not have enough information to determine the resources on which the actions are executed in our case study. If the allocation of resources is known, then with our tool, we can observe the impact of reducing large delays along the critical path i.e. reduction of end-to-end delay.

9.5 Summary

In this chapter, we analyze the Travel Management System, a case study to evaluate our work, to calculate the global delay for each involved participant. We also discuss the benefits of our work such as the calculation of the critical path, *end-to-end* delay, *server* time needed, etc.

10. CONCLUSION AND FUTURE DIRECTIONS

In the literature, there are numerous methodologies to model the behaviour of a distributed system. Modeling paradigms such as UML, Queuing Networks, and Petri-Nets are among the more popular ones. Most of these paradigms, though, assume the basic activities in the decomposition are allocated to a single component. Typically, an activity has a single input and a single output. Performance of such systems has been analyzed in various research projects.

In practice, the behaviour of a system is often specified at a higher level of abstraction, i.e. even the basic activities are relatively large. Some of these activities may involve multiple roles. For each role, there is a starting and an ending point of executions, which we call starting and ending events.

The contributions of this thesis are the following:

1. **Collaboration Behaviour Diagrams (CBD):** We explored the possibility of using existing modeling paradigms to specify the behaviour of an activity with multiple starting and ending events. To this end, we defined the semantics of the dynamic behavior of Collaborations using the formalism of partial orders, called Collaboration Behaviour Diagrams. This helped us not only to model collaborations with multiple independent starting points of execution (*starting events*) and multiple independent ending points of executions (*ending events),* but also allowed us to model the dependencies between these events.

2. **Testing Methodology:** We realized that the behaviour and the dependencies will not always be specified, and the given system description may be a black-box specification. Based on an existing testing methodology, we developed a method to measure the performance characteristics related to the dependencies amongst the starting and ending events of the roles involved in a black-box activity.
3. **Performance Analysis:** Using Collaboration Behaviour Diagrams (CBD), we defined the semantics of composite collaborations and developed a set of formulas to calculate their performance based on the delays of the constituent sub-collaborations. These compositions may consist of sub-collaborations related by strict sequence, weak sequence, concurrency, alternative executions, or strict or weak while loops.
4. **Types of Delays and Control Flow Paths:** We provide formulas to calculate different types of delays: fixed delays, range of delays and stochastic delays. Fixed and range of delays are typically used in hard real-time systems, where performance is a critical property of the system. Stochastic delays are delays that are represented by distributions such as Normal, Exponential, Poisson Distributions etc. We calculate fixed delays for a particular single control flow path while analysis for range of delays, and stochastic delays requires all possible paths.
5. **Approximation of Normal Distribution:** We propose approximations for the case of stochastic behavior with Normal distributions. We present approximations for the cases where the curves of two normal distributions are far apart and have very *little* overlap, and where the curves are close together and have a *strong* overlap. These approximations are illustrated through examples.

6. **Expected Errors:** We discuss the expected errors that may be introduced due to ignoring shared resources such as CPUs, database, etc. or ignoring dependencies in the case of stochastic behaviours.
7. **Messages:** As the collaborations we consider may involve multiple components, we consider the use of coordination messages from [9] to realize the ordering relationships among different sub-collaborations in a distributed environment. For the case that these messages have an appreciable delay we modify our performance analysis formulas to include the delay of these coordination messages.
8. **Performance Analysis Tool:** We constructed the Performance Analysis Tool, which takes as input the dynamic behaviour of a collaboration in the form of an annotated UML Activity Diagram. It analyzes the performance of the system for a single or for multiple control flow paths and calculates the delays between the different starting and ending events of the collaboration in terms of fixed delays, range of delays or stochastic delay distributions.
9. **Case Study:** We presented a fictitious case study of a Travel Management System and demonstrated the performance analysis of the global collaboration based on the performance characteristics of the sub-collaborations.

Further work could be done in this area along the following lines:

- Apply the tool for performance analysis of industrial examples.
- We would like to analyze the "Cancel/Interruption" sequencing operator and develop formulas to calculate the performance of collaborations that uses this operator.

- There were some conditions made for the sharing of resources as discussed in Chapter 7. We discussed estimates of errors introduced for some of these assumptions but not for all. Further analysis of the errors introduced by the remaining assumptions would be quite useful.
- We have made the assumption that the stochastic distributions of the performance parameters of a given sub-collaborations and of different sub-collaborations are independent. We would like to be able to relax these constraints and analyze the behaviour of a system in more general situations.
- We impose certain conditions when we approximate calculations with Normal Distributions. We would like to relax these conditions and be able to present the approximations with as little conditions as possible. Furthermore, we would like to have an estimate of the errors introduced with these approximations.
- For the calculations involving stochastic delays, we have approximated only Normal Distributions. We would like to approximate calculations for collaborations with other distributions such as Exponential, Binomial, etc.
- Finally, we would like to explore whether we could adapt our work to calculate other global measures such as cost.

REFERENCES

[1] Ajmone-Marsan, M. "Stochastic Petri nets: an elementary introduction." In Advances in Petri Nets, 1989, pp. 1-29.

[2] Amazon, http://aws.amazon.com/ec2/, accessed on April 16, 2014.

[3] Amsden J., Gardner T., Griffin C., Iyengar S., Draft UML 1.4 profile for automated business processes with a mapping to BPEL 1.0, Version 1.1, see http://www.ibm.com/developerworks/rational/library/4593.html, accessed April 16, 2014.

[4] Babka, V., Decky, M., Tuma, P. “Resource Sharing in Performance Models” in Proceedings of European Performance Engineering Workshop 2007, Springer-Verlag LNCS 4748, Jul 2007.

[5] Baier C, Boudewijn R. H., Holger H., and Joost-Pieter K, “ Model checking meets performance evaluation” ACM SIGMETRICS Performance Evaluation Review, v.32 n.4, March 2005, p.10-15.

[6] Baier C, Boudewijn R. H., Holger H., and Joost-Pieter K. "Performance evaluation and model checking join forces." Communications of the ACM v. 53, no. 9, 2010, p 76-85.

[7] Balbo, G. "Introduction to stochastic Petri nets." Lecture Notes in Computer Science Volume 2090, Springer Heidelberg, 2001, pp 84-155.

[8] Bochmann, G.v., “Deriving component designs from global requirements”, in Proceedings of International Workshop on Model Based Architecting and Construction of Embedded Systems (ACES), Toulouse, 2008, pp 55-69.

[9] Bochmann, G.v., “A General Transition Model of Protocols and Communication Services”, IEEE Transactions on Communications COM-28, no 4, April 1980, pp 643-650.

[10] Bochmann, G.v., Gotzhein, R., “Deriving protocol specifications from service specifications.” in Proceedings of Association for Computing Machinery (ACM) SIGCOMM Symposium, vol 16, no 3, 1986, pp 148-156.

[11] Bochmann, G.v., Haar, S., Jard, C., Jourdan, G.V., “Testing systems specified as partial order input/output automata.” in joint 20th IFIP TC6/WG6.1 International Conference on Testing of Software and Communicating Systems, TestCom’08, and 8th International Workshop on Formal Approaches to Software Testing, FATES’08, LNCS 5047, Springer (2008), pp. 169–183.

[12] Breza, M., lecture notes on Constructing and Solving Markov Processes, see http://www.doc.ic.ac.uk/~mjb04/markov.pdf, accessed on April 16, 2014.

[13] Casselman, R.S., Use Case Maps for Object-oriented Systems, Prentice Hall, New Jersey, 1995

[14] Castejón H., Bræk R., and Bochmann G.v., "Realizability of Collaboration-based Service Specifications". Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC 2007), IEEE Computer Society, December 2007

[15] Castejon, H., Bochmann G., Braek R., On the Realizability of Collaborative Services, Journal of Software and Systems Modeling, Vol. 10, 12 October 2011, pp. 1-21.

[16] Chapman N, Lecture Series on Petri Net Models, see

http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol2/njc1/, accessed on April 16, 2014.

[17] Chuang, L., Yang Q., Fengyuan R., Marinescu D., "Performance equivalent analysis of workflow systems based on stochastic petri net models." Engineering and Deployment of Cooperative Information Systems. 2002. 64-79.

[18] Chinneck, J., "Practical Optimization: a Gentle Introduction", online textbook, (2000), see http://www.sce.carleton.ca/faculty/chinneck/po.html, accessed on April 10, 2008.

[19] Clarke E., Emerson E., Sistla A., “Automatic verification of finite state concurrent systems using temporal logic specifications." Association for Computing Machinery (ACM) Trans on Prog Languages & Systems, Vol. 8, No. 2, April 1986, pp. 244-263.

[20] Deshpande, Jayant V., "On Continuity of a Partial Order". Proceedings of the American Mathematical Society, 1968

[21] Emerson E.A., “Temporal and modal logic”, Handbook of Theoretical Computer Science, Chapter 16, the MIT Press, 1990

[22] Esparza J., Heljanko K., Unfoldings - A Partial-Order Approach to Model Checking, New York: Springer-Verlag, 2008

[23] Fazar, W., "Program Evaluation and Review Technique", The American Statistician, Vol. 13, No. 2, p.10, 1959

[24] Faleh M.N.M., Bochmann, Gv, “Transforming dynamic behavior specifications from Activity Diagrams to BPEL”, in Service Oriented System Engineering (SOSE), 2011 IEEE 6th International Symposium on, pp. 305-311. IEEE, 2011.

[25] Foroutan S., Akesson B., Goossens K., Petrot F., "A General Framework for Average-Case Performance Analysis of Shared Resources." Digital System Design (DSD), 2013 Euromicro Conference on. IEEE, 2013.

[26] Goseva-Popstojanova K, Trivedi K.S., “Stochastic Modeling Formalisms for Dependability, Performance and Performability,” Performance Evaluation – Origins and Directions, 2000, pp. 385-404,

[27] Grabiec B., Traonouez L., Jard C., Lime D and Roux O.H., “Diagnosis using unfoldings of parametric time petri nets.” in 8th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2010), September 2010, Vienna, Austria. pp 137-151

[28] Graham, R. L.; Knuth, D. E., Patashnik, O.,Concrete Mathematics: A Foundation for Computer Science (2nd Edition)., Addison-Wesley Professional., 1994

[29] Haas J P, Stochastic Petri Nets – Modelling, Stability, Simulation, Springer-Verlag, 2002

[30] Hao, J., Pei-an W., "An approach for workflow performance evaluation based on discrete stochastic Petri net." e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on. IEEE, 2007.

[31] Hazewinkel, M., "Normal Distribution", Encyclopedia of Mathematics, Kluwer, USA, 2001

[32] He Z., Huang A., "Approximating the Minimum Distribution of Two Normally Distributed Variables Each with the Same Mean and Variance." Computational

Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on. IEEE, 2012.

[33] Hill, J. The minimum of n independent normal distributions., Technical report, http://www.untruth.org/~josh/math/normal-min.PDF, accessed on August 2011.

[34] Hogg R., Craig A., McKean J, "Introduction to Mathematical Statistics" , 6th Ed., New Jersey: Prentice Hall. p. 692, 2004

[35] Israr T., Bochmann G.v., Performance Modeling of Distributed Collaboration Services, Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering, Karlsruhe, Germany, 2011, pp 475-480

[36] Israr T., Bochmann G.v., Performance modeling of distributed collaboration services with independent inputs-outputs, Proceedings of the 5th Intern. Workshop on Non-functional Properties in Modeling: Analysis, Languages and Processes co-located with 16th Intern. Conf. on Model Driven Engineering Languages and Systems, Miami, USA, September 29, 2013

[37] Israr T., Bochmann G.v., Stochastic performance analysis of distributed activities, Proc. of 5th Intern. Workshop on Non-functional Properties in Modeling: Analysis, Languages and Processes co-located with 16th Intern. Conf. on Model Driven Engineering Languages and Systems, Miami, USA, 2013

[38] Kant K., Sundaram C. R. M., "A Server Performance Model for Static Web Workloads" Proceedings of the 2000 IEEE International Symposium on Performance Analysis of Systems and Software, p.201-206, 2000

[39] Korherr, B., Presentation on Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures, see http://www.wit.at/people/korherr/pres/Pres_ER06.PDF, accessed on April 16, 2014.

[40] Korherr, B., List, B., 2006. "Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL", Proceedings of the 2nd International Workshop on Best Practices of UML (ER 2006), Tucson, Arizona, USA, 2006.

[41] Kounev, S. "Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets" IEEE Transactions on Software Engineering, v.32 n.7, p.486-502, July 2006

[42] Kounev S. and Buchmann A., "Performance Modelling of Distributed E-Business Applications Using Queueing Petri Nets," Proc. 2003 IEEE International Symposium. Performance Analysis of Systems and Software, 2003.

[43] Kruger I., Gupta D., Mathew R., Moorthy P., Phillips W., Rittmann S., Ahluwalia J., "Towards a process and tool-chain for service-oriented automotive software engineering." in Proceedings of the 26th international Conference on Software Engineering, Workshop on Software Engineering for Automotive Systems (SEAS), 2004

[44] Ku, T, Lecture Notes, http://www.ee.hawaii.edu/~kuh/ee342.f10/sumsofrvs.PDF, accessed on January 12, 2014

[45] Laamarti F., Derivation of component designs from a global specification, MSc thesis,

University of Ottawa, 2010. See http://www.site.uottawa.ca/~bochmann/dsrg/PublicDocuments/Mastertheses/Laamarti%20-%20Derivation-of-Component-Designsfrom-Global-Specifications.PDF, accessed on April 16, 2014

[46] Lamport, L., "Time, clocks, and the ordering of events in a distributed system", Communications of ACM, pp. 558-565, 1978

[47] Lazowska E., Zahorjan J., Graham G., Sevcik K., Quantitative system performance: computer system analysis using queuing network models, Prentice-Hall, Inc., Upper Saddle River, NJ, 1984

[48] Li J., Fan Y., and Zhou M. "Performance modeling and analysis of workflow." In Proceedings of IEEE Transitions of Systems, Man and Cybernetics - Part A: Systems and Humans, 229-242, 2004

[49] Liu Y., Fekete A., Gorton I. "Predicting the Performance of Middleware-Based Applications at the Design Level." in: Proceedings of the Fourth International Workshop on Software and Performance 2004, , Redwood Shores, California, USA, 2004,

[50] Liu Y., Gorton I. "Performance Prediction of J2EE Applications Using Messaging Protocols", Proceeding of International SIGSOFT Symposium - Component-Based Software Eng. (CBSE',05), pp. 1-16, 2005.

[51] Luo G., Dssouli R., Bochmann G.v., Venkataram P., Ghedamsi A., "Generating Synchronizable Test Sequences Based on Finite State Machine with Distributed Ports",

Proceedings of the IFIP TC6/WG6.1 Sixth International Workshop on Protocol Test systems VI, p.139-153, 1993

[52] Mantell, From UML to BPEL, see http://www.ibm.com/developerworks/webservices/library/ws-uml2bpel/, accessed on April 16, 2014

[53] MARTE Profile, see http://www.omgmarte.org, accessed on April 16, 2014

[54] Mayer P., Schroeder A., and Koch N., "A Model-Driven Approach to Service Orchestration". Intl. Conference on Services Computing (SCC), Honolulu, USA, 2008.

[55] McMillan K.L., “Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronous Circuits.”, in Proceedings of the Fourth International Workshop on Computer Aided Verification, 1992, p.164-177

[56] McNeile, A., “Using Motivation and Choreography to model Distributed Workflow” Proceedings of the 5th ACM SIGCHI Annual International Workshop on Behaviour Modelling- Foundations and Applications, Montpellier, France, 2013

[57] Merlin P., Faber D., “Recoverability of communication protocols – Implications of a Theoretical Study”, IEEE Transaction Communication, vol. COM-24, no. 9, Sept. 1976.

[58] Meeker W.Q., Cornwell L.W., Aroian L.A., The product of two normally distributed random variables. American Mathematical Society, 1981.

[59] Naumovich G., Clarke L. and Cobleigh J., “Using partial order techniques to improve performance of data flow analysis based verification.” in Proceedings of the 1999 ACM

SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, p.57-65, 1999, Toulouse, France

[60] OASIS BPEL. 2010. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, accessed on April 16, 2014.

[61] Object Management Group. UML Profile for Schedulability, Performance and Time Specification, January 2005. V1.1, f/05-01-02.

[62] Object Management Group, UML 2.0 Superstructure Specification. http://www.omg.org/cgi-bin/doc?ptc/2003-08-02., accessed on April 16, 2014.

[63] Ouyang C., Aalst Q., Dumas M., and Hofstede A. "Translating BPMN to BPEL.", Technical Report BPM-06-02, BPMcenter.org, 2006.

[64] Owen M, Raj J, Introduction to the New Business Process Modeling Standard, see http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.PDF, accessed on April 16, 2014.

[65] Panayiotou C, lecture slides on Generalized Semi-Markov Processes, see http://www.eng.ucy.ac.cy/christos/courses/ECE658/Lectures/GSMP.ppt, accessed on April 16, 2014.

[66] Patel, J. K. and Read, C. B. Handbook of the Normal Distribution. New York: Dekker, 1982.

[67] Pnueli, A., The temporal logic of programs. Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS), 1977, 46–57.

[68] Pressman, Software Engineering: A Practitioner's Approach, Prentice-Hall, 1996.

[69] Program Evaluation and Review Technique, see http://www.netmba.com/operations/project/pert/, accessed April 16, 2014.

[70] Rational Software Architect, See http://www-306.ibm.com/software/awdtools/architect/swarchitect/, accessed April 16, 2014.

[71] Razouk, The Derivation of Performance Expressions for Communication Protocols from Timed Petri Net models, Association for Computing Machinery (ACM) SIGCOMM Computer Communication Review, v.14 n.2, pp 210-217, June 1984

[72] Roßler, Geppert, Gotzhein: Collaboration-based design of SDL systems. In: 10th SDL Forum. LNCS 2078 (2001) 72–89

[73] Rozenburg G., Engelfriet J., Elementary Net Systems, in Lectures on Petri Nets I: Basic Models - Advances in Petri Nets, volume 1491 of Lecture Notes in Computer Science, Springer, pp. 12-121, 1998.

[74] Ryan K. L. K., Stephen S. G. L., Eng W. L., "Business Process Management (BPM) Standards: A Survey." in: Business Process Management Journal, Emerald Group Publishing Limited. Volume 15 Issue 5, 2009

[75] Sahner R., Trivedi K., Performance and reliability analysis using directed acyclic graphs. IEEE Trans. Software Eng., pp. 1105–1114, 1987

[76] Shapiro, Gagne, "What is new in XPLD", http://www.wfmc.org/Download-document/Whats-New-in-XPDL-2.2.html, accessed April 16, 2014.

[77] Sheldon M R., "Expectation of a random variable". Introduction to probability models (9th ed.). Academic Press. p. 38, 2007.

[78] Sifakis, J. "Use of Petri Nets for performance evaluation"; in Measuring, modeling and evaluating computer systems, North-Holland 1977, pp 75-93

[79] Skiena, S. "Hasse Diagrams." in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Addison-Wesley, p. 163, 169-170, and 206-208, 1990.

[80] Smith, J.O., Product of Two Gaussian PDFs, https://ccrma.stanford.edu/~jos/sasp/Product_Two_Gaussian_PDFs.html, online book, accessed April 16, 2014.

[81] Smith U.C., Performance Engineering of Software Systems, Addison-Wesley, 2002

[82] Symons, FJW. Introduction to numerical Petri nets, a general graphical model of concurrent processing systems. Australian Telecommunications Research, 14(1):28–33, January 1980

[83] W3C, Web Services Choreography Description Language (WS-CDL), Version 1.0, December 2004

[84] Wang Y., Lin C., Ungsunan P. Huang X., "Modeling and survivability analysis of service composition using Stochastic Petri Nets." The Journal of Supercomputing 56.1 79-105, 2011.

[85] Weisstein, E. W. "Normal Product Distribution." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/NormalProductDistribution.html, accessed on April 16, 2014.

[86] White, S., Introduction to BPMN, see

http://www.bpmn.org/Documents/Introduction_to_BPMN.PDF, accessed on April 16, 2014.

[87] Whitt, Open and closed models for network of queues, AT&T Technical Journal, 63 (9), pp. 1911–1979, 1984

[88] Wilde, Lecture series on Business Process Execution Language, http://dret.net/lectures/services-fall06/bpel2, accessed on March 2, 2009.

[89] Wolper, and Godefroid, Partial-order Methods for Temporal Verification. In Proc. 4th Int. Conference on Concurrency Theory (CONCUR), volume 715 of Lecture Notes in Computer Science, Springer, 1993. pp 233–246

[90] Woodside, Resource Page on Layered Queuing, see http://www.layeredqueues.org/, accessed on August 14, 2010.

[91] Wu and Woodside, "An Aggregation Approach to Constructing Hybrid Layered Queuing Models", in 7th Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS7), Torino, Italy, September 2005.

[92] Wu X. P., Woodside C. M.: Performance Modeling from Software Components, WOSP'04, ACM 2004

[93] Xu J., Oufimtsev A., Woodside C. M., Murphy L.: Performance Modeling and Prediction of Enterprise JavaBeans with Layered Queuing Network Templates, SIG-SOFT SEN 31(2), ACM 2006

[94] Xu J.,Woodside C. M.: Template-Driven Performance Modeling of Enterprise Java Beans, MWS'05, IEEE 2005

ANNEX A

For the Travel Management System case study, we first constructed an annotated activity model in Rational Software Architect, as shown in Figure 36 below. In this model, the local NETDs specified between the starting and ending events of each role for each activity is shown in Table 18. This annotated model is imported into our tool. Our tool analyzes this model and calculates the global NETD, as shown in Tables 19 and 20 below. In this example, we calculated fixed delays, and due to a deterministic control flow, Table 19 shows the delays when collaboration "L" was executed and Table 20 shows the global NETD when sub-collaboration "EI" was executed.

A.1 Input

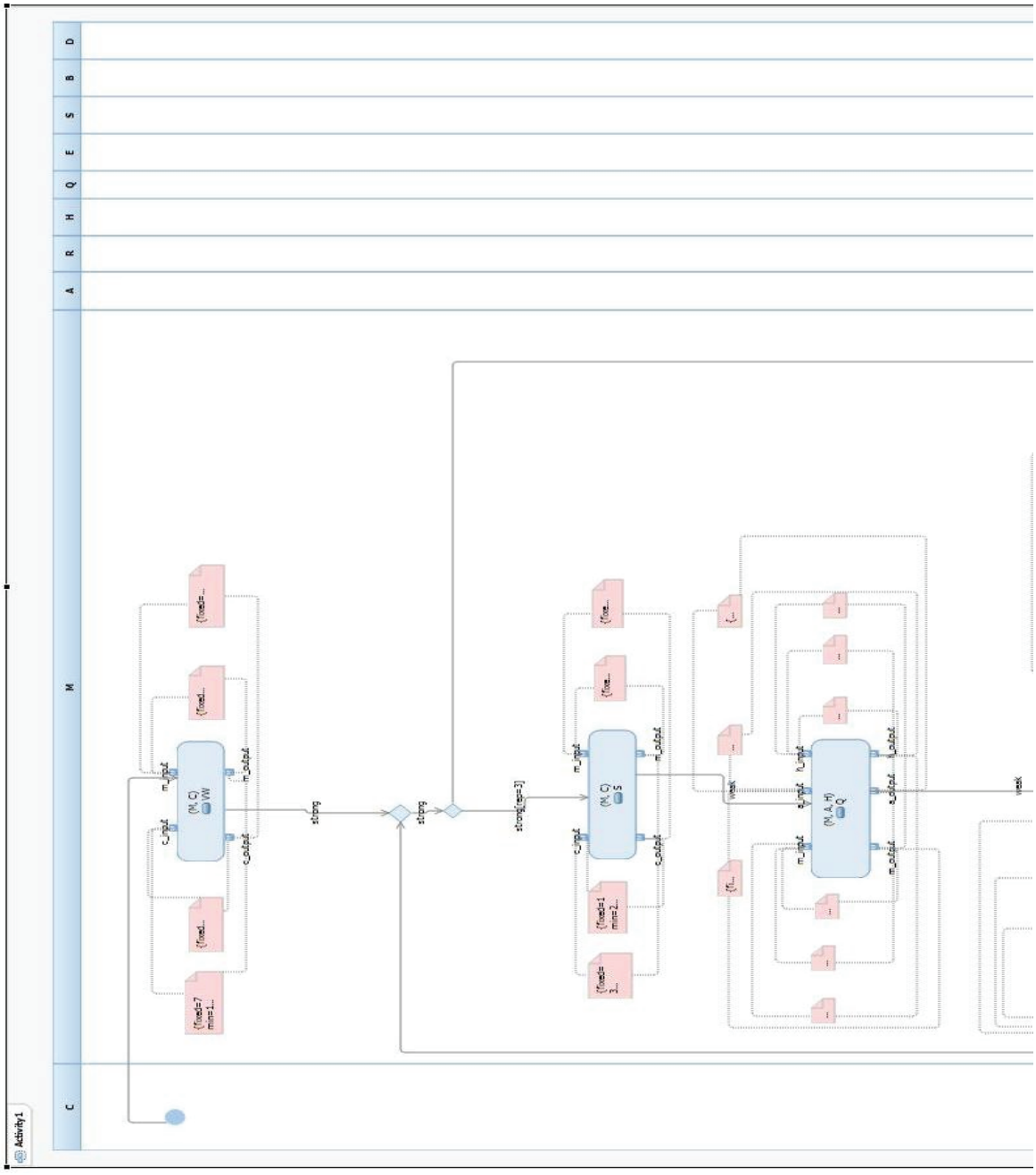


Figure 36a – Travel Management System (Part 1)

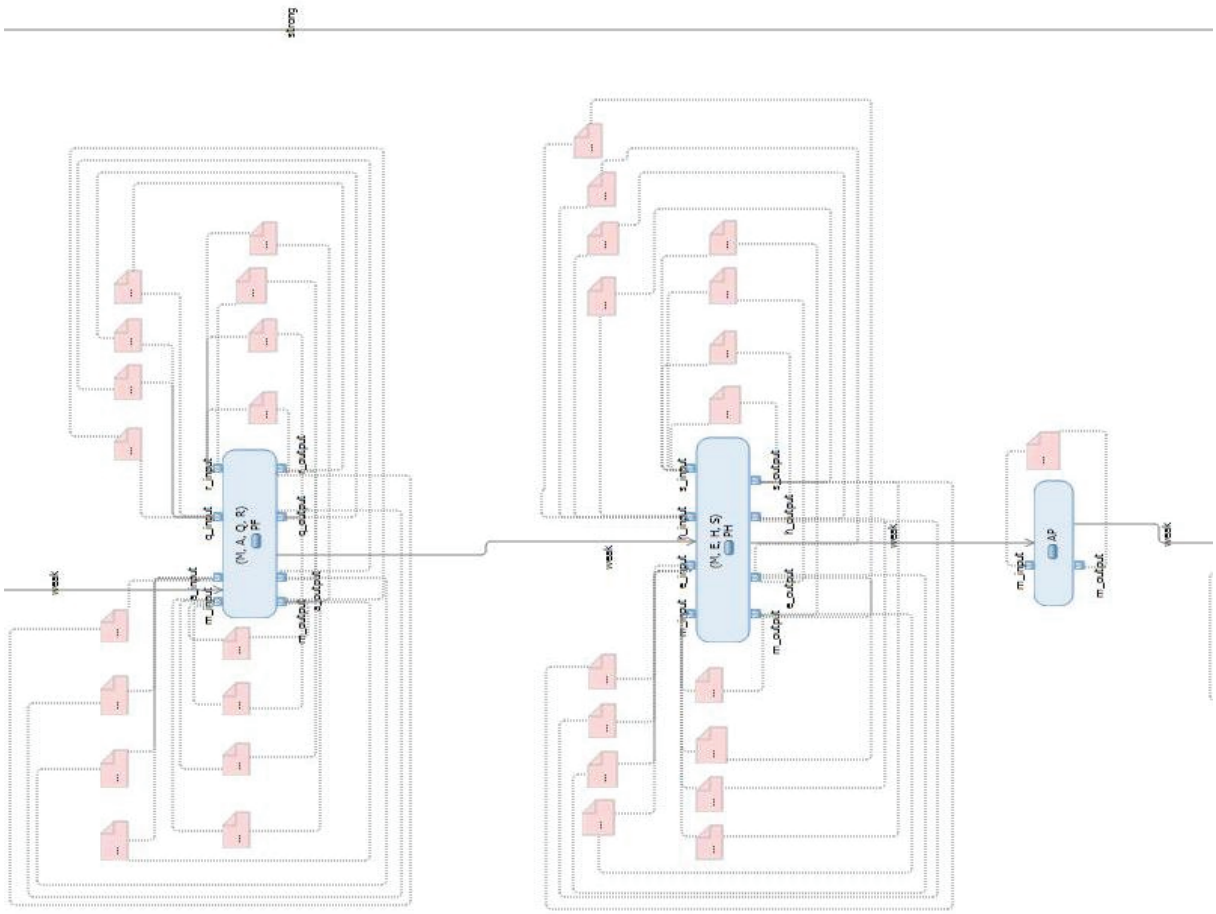


Figure 36b – Travel Management System (Part 2)

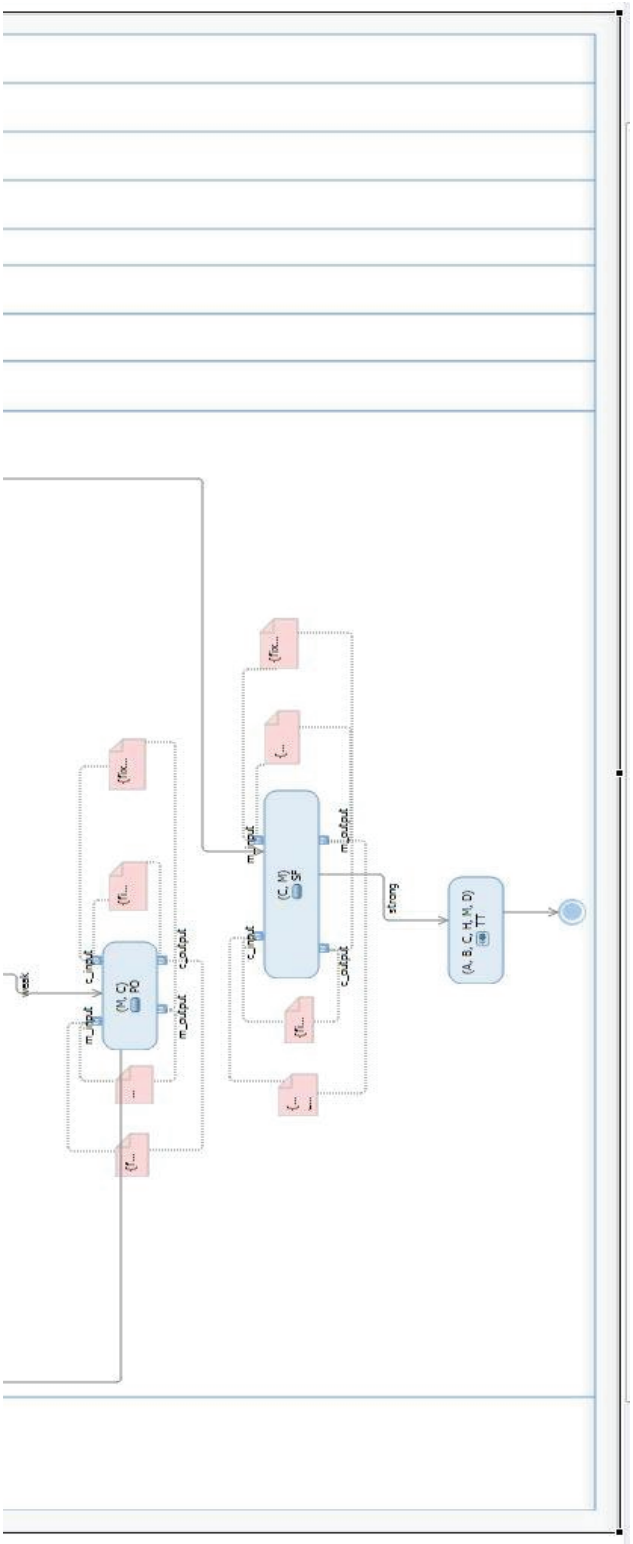


Figure 36c – Travel Management System (Part 3)

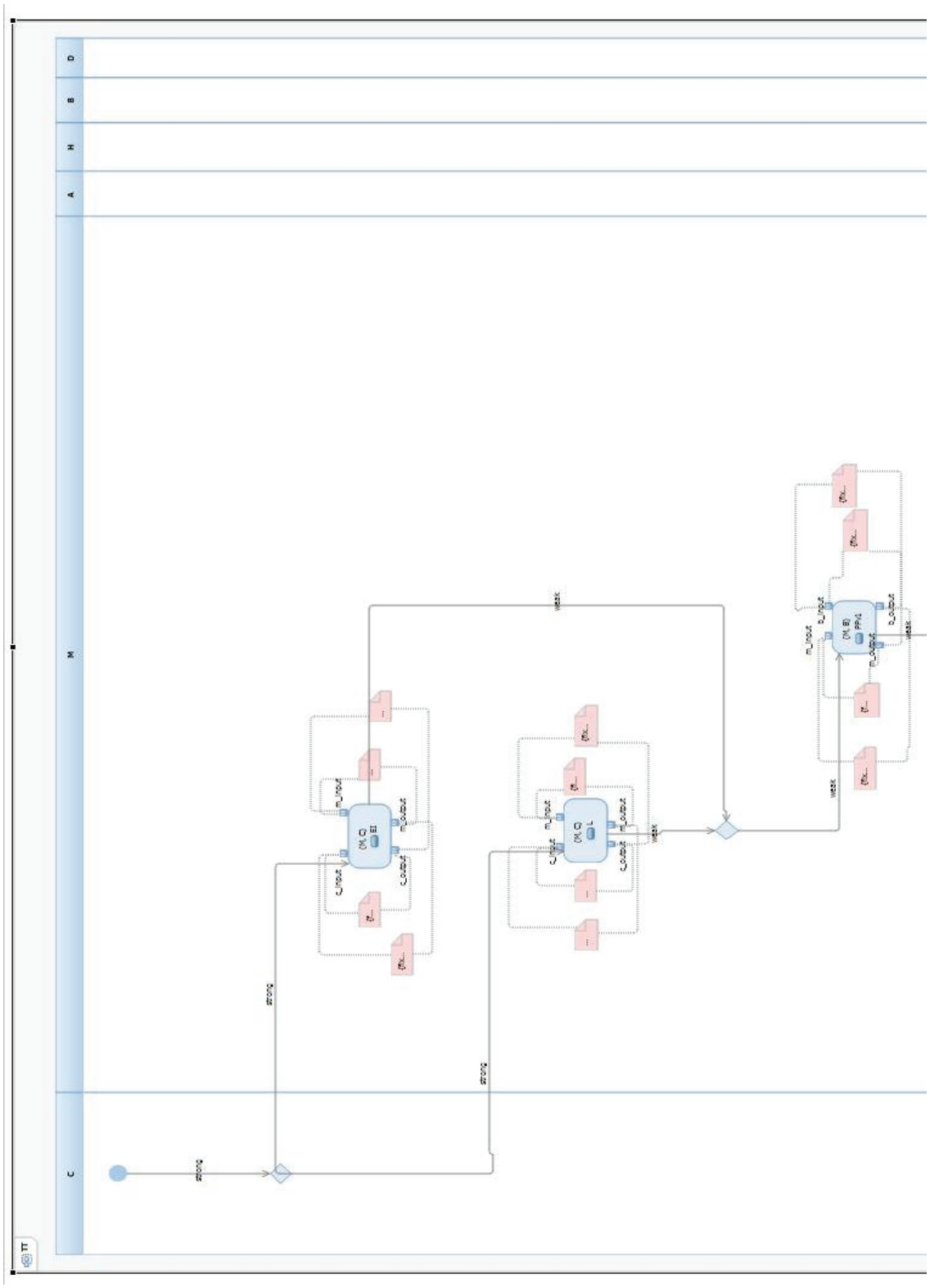


Figure 37a – Travel Management System – Nested Activity (Part 1)

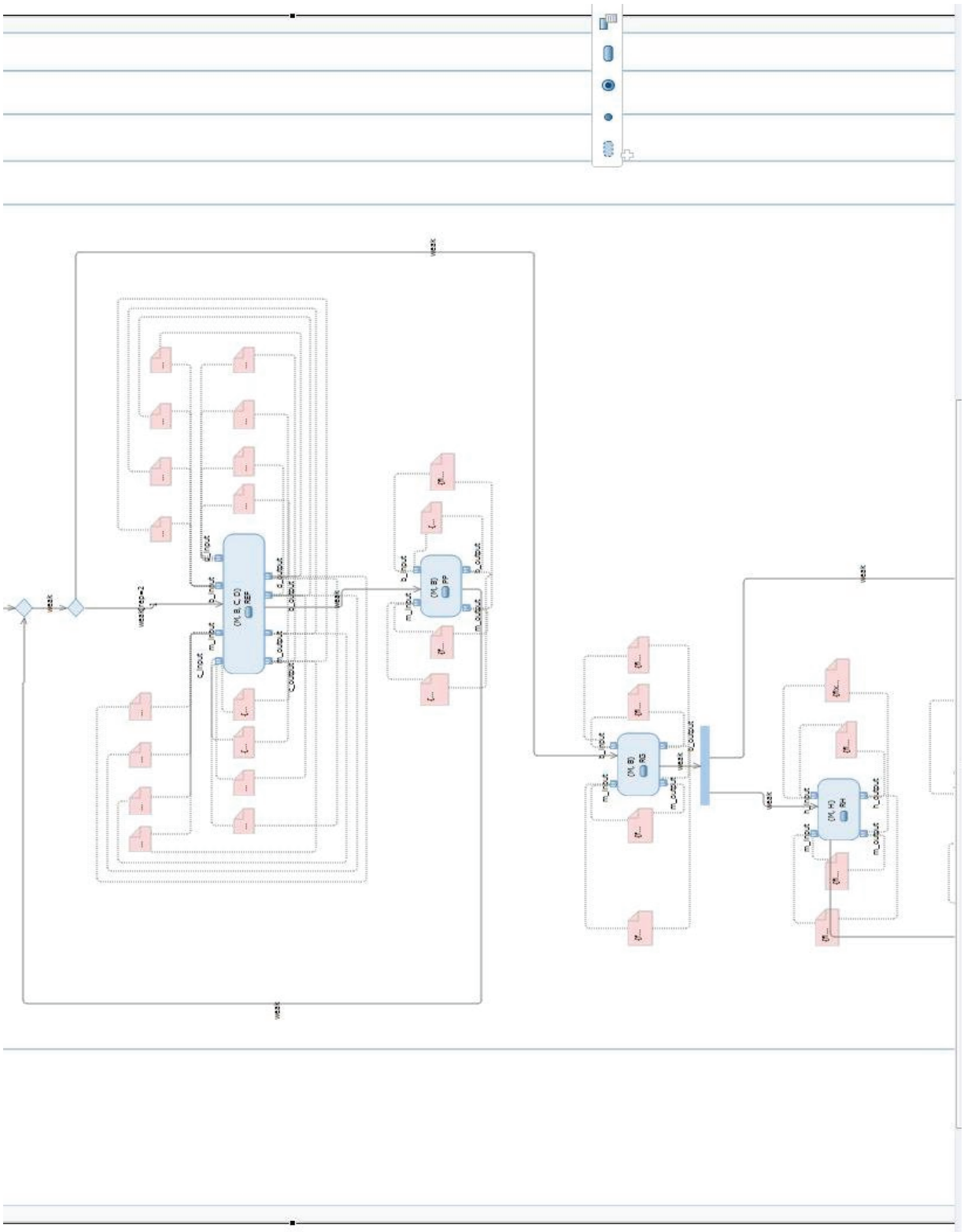


Figure 37b – Travel Management System – Nested Activity (Part 2)

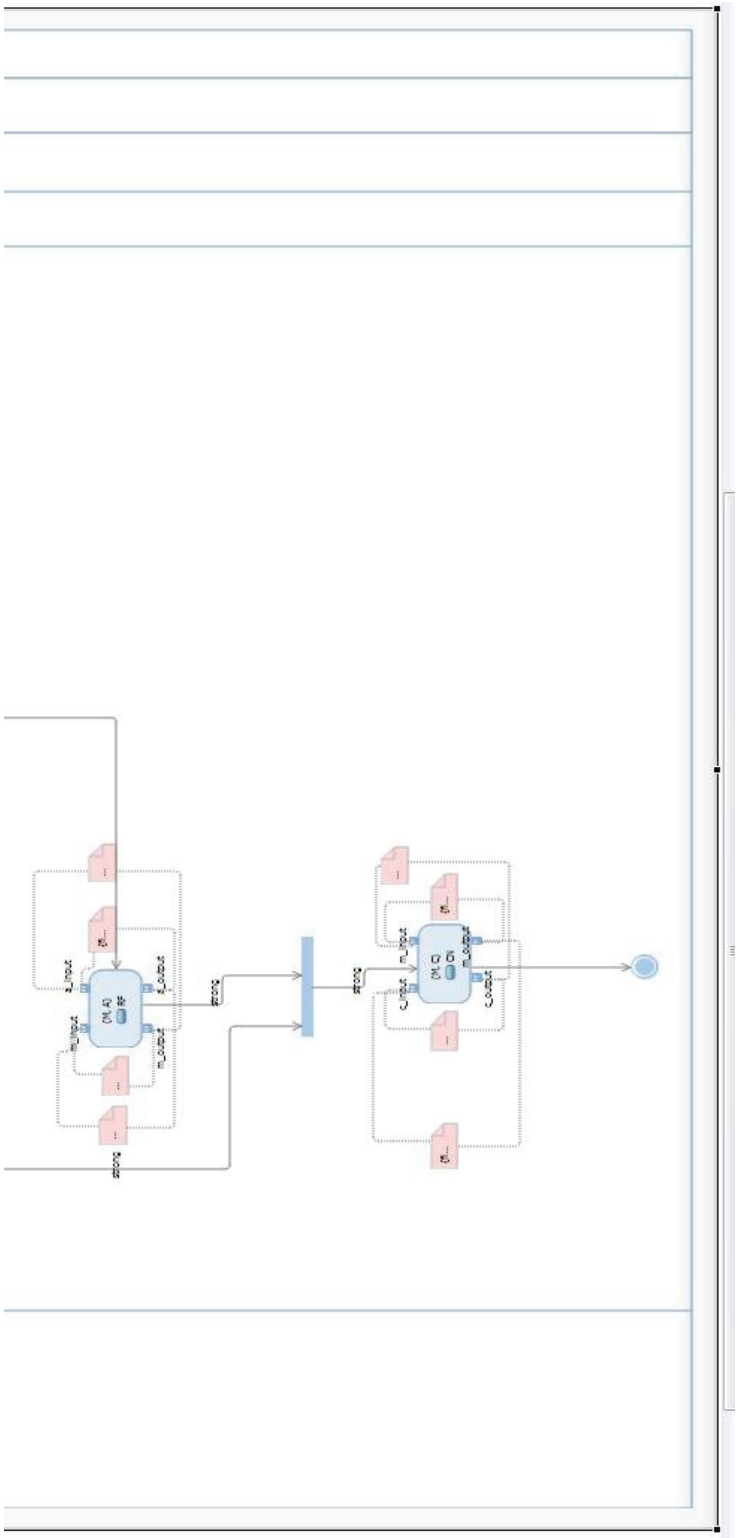


Figure 37c – Travel Management System – Nested Activity (Part 3)

A.2 Nominal Execution Time Delays (in secs) for the Travel Management Service

Table 18 – Local NETDs

	Delay		Delay		Delay		Delay		Delay
Visit Website (V):									
${}_{V}\Delta^{C}_{C}$	6	${}_{V}\Delta^{C}_{M}$	7	${}_{V}\Delta^{M}_{M}$	2	${}_{V}\Delta^{M}_{C}$	3		
Search (S):									
${}_{S}\Delta^{C}_{C}$	1	${}_{S}\Delta^{C}_{M}$	3	${}_{S}\Delta^{M}_{M}$	5	${}_{S}\Delta^{M}_{C}$	2		
Query (Q):									
${}_{Q}\Delta^{M}_{M}$	3	${}_{Q}\Delta^{M}_{A}$	2	${}_{Q}\Delta^{M}_{H}$	1	${}_{Q}\Delta^{A}_{M}$	6	${}_{Q}\Delta^{A}_{A}$	5
${}_{Q}\Delta^{A}_{H}$	4	${}_{Q}\Delta^{H}_{M}$	3	${}_{Q}\Delta^{H}_{A}$	2	${}_{Q}\Delta^{H}_{H}$	1		
Process Flights (PF):									
${}_{PF}\Delta^{M}_{M}$	7	${}_{PF}\Delta^{M}_{A}$	2	${}_{PF}\Delta^{M}_{Q}$	5	${}_{PF}\Delta^{M}_{R}$	4	${}_{PF}\Delta^{A}_{M}$	6
${}_{PF}\Delta^{A}_{A}$	2	${}_{PF}\Delta^{A}_{Q}$	3	${}_{PF}\Delta^{A}_{R}$	6	${}_{PF}\Delta^{Q}_{M}$	1	${}_{PF}\Delta^{Q}_{A}$	5
${}_{PF}\Delta^{Q}_{Q}$	6	${}_{PF}\Delta^{Q}_{R}$	4	${}_{PF}\Delta^{R}_{M}$	5	${}_{PF}\Delta^{R}_{A}$	3	${}_{PF}\Delta^{R}_{Q}$	4
${}_{PF}\Delta^{R}_{R}$	6								
Process Hotels (PH):									
${}_{PF}\Delta^{M}_{M}$	5	${}_{PF}\Delta^{M}_{H}$	6	${}_{PF}\Delta^{M}_{E}$	3	${}_{PF}\Delta^{M}_{S}$	2	${}_{PF}\Delta^{H}_{M}$	2
${}_{PF}\Delta^{H}_{H}$	6	${}_{PF}\Delta^{H}_{E}$	7	${}_{PF}\Delta^{H}_{S}$	2	${}_{PF}\Delta^{E}_{M}$	4	${}_{PF}\Delta^{E}_{H}$	1
${}_{PF}\Delta^{E}_{E}$	6	${}_{PF}\Delta^{E}_{S}$	5	${}_{PF}\Delta^{S}_{M}$	2	${}_{PF}\Delta^{S}_{H}$	3	${}_{PF}\Delta^{S}_{E}$	6
${}_{PF}\Delta^{S}_{S}$	5								
Append Profits (AP):									
${}_{PF}\Delta^{M}_{M}$	5								
Present Options (PO):									
${}_{PO}\Delta^{C}_{C}$	7	${}_{PO}\Delta^{C}_{M}$	1	${}_{PO}\Delta^{M}_{M}$	6	${}_{PO}\Delta^{M}_{C}$	3		
Select Flight & Hotel (SF)									
${}_{SF}\Delta^{C}_{C}$	7	${}_{SF}\Delta^{C}_{M}$	6	${}_{SF}\Delta^{M}_{M}$	2	${}_{SF}\Delta^{M}_{C}$	3		

Enter Information (EI):									
${}_{EI}\Delta^{C}{}_{C}$	2	${}_{EI}\Delta^{C}{}_{M}$	3	${}_{EI}\Delta^{M}{}_{M}$	4	${}_{EI}\Delta^{M}{}_{C}$	6		
Login (L):									
${}_{L}\Delta^{C}{}_{C}$	4	${}_{L}\Delta^{C}{}_{M}$	2	${}_{L}\Delta^{M}{}_{M}$	2	${}_{L}\Delta^{M}{}_{C}$	3.5		
Process Payment (PP):									
${}_{PP}\Delta^{M}{}_{M}$	2	${}_{PP}\Delta^{M}{}_{B}$	7	${}_{PP}\Delta^{B}{}_{M}$	1	${}_{PP}\Delta^{B}{}_{B}$	3		
Re-Enter Financials (REF):									
${}_{REF}\Delta^{C}{}_{C}$	2	${}_{REF}\Delta^{C}{}_{M}$	3	${}_{REF}\Delta^{C}{}_{B}$	1	${}_{REF}\Delta^{C}{}_{D}$	6	${}_{REF}\Delta^{M}{}_{C}$	3
${}_{REF}\Delta^{M}{}_{M}$	6	${}_{REF}\Delta^{M}{}_{B}$	5	${}_{REF}\Delta^{M}{}_{D}$	4	${}_{REF}\Delta^{B}{}_{C}$	5	${}_{REF}\Delta^{B}{}_{M}$	6
${}_{REF}\Delta^{B}{}_{B}$	3	${}_{REF}\Delta^{B}{}_{D}$	2	${}_{REF}\Delta^{D}{}_{C}$	4	${}_{REF}\Delta^{D}{}_{M}$	3	${}_{REF}\Delta^{D}{}_{B}$	2
${}_{REF}\Delta^{D}{}_{D}$	1								
Receipt Generated (RG):									
${}_{RG}\Delta^{M}{}_{M}$	3	${}_{RG}\Delta^{M}{}_{B}$	1	${}_{RG}\Delta^{B}{}_{M}$	1.5	${}_{RG}\Delta^{B}{}_{B}$	2		
Reserve Hotel (RH):									
${}_{RH}\Delta^{M}{}_{M}$	6	${}_{RH}\Delta^{M}{}_{H}$	1	${}_{RH}\Delta^{H}{}_{M}$	2	${}_{RH}\Delta^{H}{}_{H}$	5		
Reserve Flight (RF):									
${}_{RF}\Delta^{M}{}_{M}$	1	${}_{RF}\Delta^{M}{}_{A}$	3	${}_{RF}\Delta^{A}{}_{M}$	6	${}_{RF}\Delta^{A}{}_{A}$	4		
Confirmation Notice (CN):									
${}_{CN}\Delta^{C}{}_{C}$	3	${}_{CN}\Delta^{C}{}_{M}$	4	${}_{CN}\Delta^{M}{}_{M}$	1	${}_{CN}\Delta^{M}{}_{C}$	2		

A.3 Output Generated by the Tool

Table 19 – Global NETD (Fixed Delay) when Sub-collaboration "L" is Executed

	Delay
${}_{D}\Delta^{C}_{C}$	127.0
${}_{D}\Delta^{C}_{M}$	128.0
${}_{D}\Delta^{C}_{A}$	121.0
${}_{D}\Delta^{C}_{H}$	119.0
${}_{D}\Delta^{C}_{Q}$	73.0
${}_{D}\Delta^{C}_{R}$	73.0
${}_{D}\Delta^{C}_{E}$	73.0
${}_{D}\Delta^{C}_{S}$	73.0
${}_{D}\Delta^{C}_{B}$	118.5
${}_{D}\Delta^{C}_{D}$	105.5
${}_{D}\Delta^{Q}_{C}$	110.0
${}_{D}\Delta^{Q}_{M}$	111.0
${}_{D}\Delta^{Q}_{A}$	104.0
${}_{D}\Delta^{Q}_{H}$	102.0
${}_{D}\Delta^{Q}_{Q}$	56.0
${}_{D}\Delta^{Q}_{R}$	56.0
${}_{D}\Delta^{Q}_{E}$	56.0

	Delay
${}_{D}\Delta^{M}_{C}$	123.0
${}_{D}\Delta^{M}_{M}$	124.0
${}_{D}\Delta^{M}_{A}$	117.0
${}_{D}\Delta^{M}_{H}$	115.0
${}_{D}\Delta^{M}_{Q}$	69.0
${}_{D}\Delta^{M}_{R}$	69.0
${}_{D}\Delta^{M}_{E}$	69.0
${}_{D}\Delta^{M}_{S}$	69.0
${}_{D}\Delta^{M}_{B}$	114.5
${}_{D}\Delta^{M}_{D}$	101.5
${}_{D}\Delta^{R}_{C}$	114.0
${}_{D}\Delta^{R}_{M}$	115.0
${}_{D}\Delta^{R}_{A}$	108.0
${}_{D}\Delta^{R}_{H}$	106.0
${}_{D}\Delta^{R}_{Q}$	60.0
${}_{D}\Delta^{R}_{R}$	60.0
${}_{D}\Delta^{R}_{E}$	60.0

	Delay
${}_{D}\Delta^{A}_{C}$	120.0
${}_{D}\Delta^{A}_{M}$	121.0
${}_{D}\Delta^{A}_{A}$	114.0
${}_{D}\Delta^{A}_{H}$	112.0
${}_{D}\Delta^{A}_{Q}$	66.0
${}_{D}\Delta^{A}_{R}$	66.0
${}_{D}\Delta^{A}_{E}$	66.0
${}_{D}\Delta^{A}_{S}$	66.0
${}_{D}\Delta^{A}_{B}$	111.5
${}_{D}\Delta^{A}_{D}$	98.5
${}_{D}\Delta^{E}_{C}$	108.0
${}_{D}\Delta^{E}_{M}$	109.0
${}_{D}\Delta^{E}_{A}$	102.0
${}_{D}\Delta^{E}_{H}$	100.0
${}_{D}\Delta^{E}_{Q}$	54.0
${}_{D}\Delta^{E}_{R}$	54.0
${}_{D}\Delta^{E}_{E}$	54.0

	Delay
${}_{D}\Delta^{H}_{C}$	117.0
${}_{D}\Delta^{H}_{M}$	118.0
${}_{D}\Delta^{H}_{A}$	111.0
${}_{D}\Delta^{H}_{H}$	109.0
${}_{D}\Delta^{H}_{Q}$	63.0
${}_{D}\Delta^{H}_{R}$	63.0
${}_{D}\Delta^{H}_{E}$	63.0
${}_{D}\Delta^{H}_{S}$	63.0
${}_{D}\Delta^{H}_{B}$	108.5
${}_{D}\Delta^{H}_{D}$	95.5
${}_{D}\Delta^{S}_{C}$	106.0
${}_{D}\Delta^{S}_{M}$	107.0
${}_{D}\Delta^{S}_{A}$	100.0
${}_{D}\Delta^{S}_{H}$	98.0
${}_{D}\Delta^{S}_{Q}$	52.0
${}_{D}\Delta^{S}_{R}$	52.0
${}_{D}\Delta^{S}_{E}$	52.0

${}_{D}\Delta^{Q}{}_{S}$	56.0
${}_{D}\Delta^{Q}{}_{B}$	101.5
${}_{D}\Delta^{Q}{}_{D}$	88.5
${}_{D}\Delta^{B}{}_{C}$	39.5
${}_{D}\Delta^{B}{}_{M}$	40.5
${}_{D}\Delta^{B}{}_{A}$	33.5
${}_{D}\Delta^{B}{}_{H}$	31.5
${}_{D}\Delta^{B}{}_{Q}$	$-\infty$
${}_{D}\Delta^{B}{}_{R}$	$-\infty$
${}_{D}\Delta^{B}{}_{E}$	$-\infty$
${}_{D}\Delta^{B}{}_{S}$	$-\infty$
${}_{D}\Delta^{B}{}_{B}$	31
${}_{D}\Delta^{B}{}_{D}$	18

${}_{D}\Delta^{R}{}_{S}$	60.0
${}_{D}\Delta^{R}{}_{B}$	105.5
${}_{D}\Delta^{R}{}_{D}$	92.5
${}_{D}\Delta^{D}{}_{C}$	33.5
${}_{D}\Delta^{D}{}_{M}$	34.5
${}_{D}\Delta^{D}{}_{A}$	27.5
${}_{D}\Delta^{D}{}_{H}$	25.5
${}_{D}\Delta^{D}{}_{Q}$	$-\infty$
${}_{D}\Delta^{D}{}_{R}$	$-\infty$
${}_{D}\Delta^{D}{}_{E}$	$-\infty$
${}_{D}\Delta^{D}{}_{S}$	$-\infty$
${}_{D}\Delta^{D}{}_{B}$	25
${}_{D}\Delta^{D}{}_{D}$	12

${}_{D}\Delta^{E}{}_{S}$	54.0
${}_{D}\Delta^{E}{}_{B}$	99.5
${}_{D}\Delta^{E}{}_{D}$	86.5

${}_{D}\Delta^{S}{}_{S}$	52.0
${}_{D}\Delta^{S}{}_{B}$	97.5
${}_{D}\Delta^{S}{}_{D}$	84.5

Table 20 – Global NETD when Sub-collaboration EI is Executed

	Delay
${}_D\Delta^C{}_C$	129.5
${}_D\Delta^C{}_M$	130.5
${}_D\Delta^C{}_A$	123.5
${}_D\Delta^C{}_H$	121.5
${}_D\Delta^C{}_Q$	73.0
${}_D\Delta^C{}_R$	73.0
${}_D\Delta^C{}_E$	73.0
${}_D\Delta^C{}_S$	73.0
${}_D\Delta^C{}_B$	121.0
${}_D\Delta^C{}_D$	108.0
${}_D\Delta^Q{}_C$	112.5
${}_D\Delta^Q{}_M$	113.5
${}_D\Delta^Q{}_A$	106.5
${}_D\Delta^Q{}_H$	104.5
${}_D\Delta^Q{}_Q$	56.0
${}_D\Delta^Q{}_R$	56.0
${}_D\Delta^Q{}_E$	56.0
${}_D\Delta^Q{}_S$	56.0
${}_D\Delta^Q{}_B$	104.0

	Delay
${}_D\Delta^M{}_C$	125.5
${}_D\Delta^M{}_M$	126.5
${}_D\Delta^M{}_A$	119.5
${}_D\Delta^M{}_H$	117.5
${}_D\Delta^M{}_Q$	69.0
${}_D\Delta^M{}_R$	69.0
${}_D\Delta^M{}_E$	69.0
${}_D\Delta^M{}_S$	69.0
${}_D\Delta^M{}_B$	117.0
${}_D\Delta^M{}_D$	104.0
${}_D\Delta^R{}_C$	116.5
${}_D\Delta^R{}_M$	117.5
${}_D\Delta^R{}_A$	110.5
${}_D\Delta^R{}_H$	108.5
${}_D\Delta^R{}_Q$	60.0
${}_D\Delta^R{}_R$	60.0
${}_D\Delta^R{}_E$	60.0
${}_D\Delta^R{}_S$	60.0
${}_D\Delta^R{}_B$	108.0

	Delay
${}_D\Delta^A{}_C$	122.5
${}_D\Delta^A{}_M$	123.5
${}_D\Delta^A{}_A$	116.5
${}_D\Delta^A{}_H$	114.5
${}_D\Delta^A{}_Q$	66.0
${}_D\Delta^A{}_R$	66.0
${}_D\Delta^A{}_E$	66.0
${}_D\Delta^A{}_S$	66.0
${}_D\Delta^A{}_B$	114.0
${}_D\Delta^A{}_D$	101.0
${}_D\Delta^E{}_C$	110.5
${}_D\Delta^E{}_M$	111.5
${}_D\Delta^E{}_A$	104.5
${}_D\Delta^E{}_H$	102.5
${}_D\Delta^E{}_Q$	54.0
${}_D\Delta^E{}_R$	54.0
${}_D\Delta^E{}_E$	54.0
${}_D\Delta^E{}_S$	54.0
${}_D\Delta^E{}_B$	102.0

	Delay
${}_D\Delta^H{}_C$	119.5
${}_D\Delta^H{}_M$	120.5
${}_D\Delta^H{}_A$	113.5
${}_D\Delta^H{}_H$	111.5
${}_D\Delta^H{}_Q$	63.0
${}_D\Delta^H{}_R$	63.0
${}_D\Delta^H{}_E$	63.0
${}_D\Delta^H{}_S$	63.0
${}_D\Delta^H{}_B$	111.0
${}_D\Delta^H{}_D$	98.0
${}_D\Delta^S{}_C$	108.5
${}_D\Delta^S{}_M$	109.5
${}_D\Delta^S{}_A$	102.5
${}_D\Delta^S{}_H$	100.5
${}_D\Delta^S{}_Q$	52.0
${}_D\Delta^S{}_R$	52.0
${}_D\Delta^S{}_E$	52.0
${}_D\Delta^S{}_S$	52.0
${}_D\Delta^S{}_B$	100.0

${}_{D}\Delta^{Q}{}_{D}$	91.0
${}_{D}\Delta^{B}{}_{C}$	39.5
${}_{D}\Delta^{B}{}_{M}$	40.5
${}_{D}\Delta^{B}{}_{A}$	33.5
${}_{D}\Delta^{B}{}_{H}$	31.5
${}_{D}\Delta^{B}{}_{Q}$	$-\infty$
${}_{D}\Delta^{B}{}_{R}$	$-\infty$
${}_{D}\Delta^{B}{}_{E}$	$-\infty$
${}_{D}\Delta^{B}{}_{S}$	$-\infty$
${}_{D}\Delta^{B}{}_{B}$	31.0
${}_{D}\Delta^{B}{}_{D}$	18.0

${}_{D}\Delta^{R}{}_{D}$	95.0
${}_{D}\Delta^{D}{}_{C}$	33.5
${}_{D}\Delta^{D}{}_{M}$	34.5
${}_{D}\Delta^{D}{}_{A}$	27.5
${}_{D}\Delta^{D}{}_{H}$	25.5
${}_{D}\Delta^{D}{}_{Q}$	$-\infty$
${}_{D}\Delta^{D}{}_{R}$	$-\infty$
${}_{D}\Delta^{D}{}_{E}$	$-\infty$
${}_{D}\Delta^{D}{}_{S}$	$-\infty$
${}_{D}\Delta^{D}{}_{B}$	25.0
${}_{D}\Delta^{D}{}_{D}$	12.0

${}_{D}\Delta^{E}{}_{D}$	89.0

${}_{D}\Delta^{S}{}_{D}$	87.0

ANNEX B – TEST CASES

We tested our tool extensively. For each sequencing operator, we considered a single control flow and multiple control flows. For a single control flow, we calculated the global NETD as fixed delays, range of delays and delay distributions (stochastic delays). For multiple control flows, we calculated NETDs as range of delays and delay distributions (stochastic delays). For each of the delays calculated, we also kept track of the path of execution required to achieve the calculated delay. The test cases are shown in Table 21 below, where each "X" represents a single test case. Each case under each delay type (i.e. 1,2,3,4..) represents the conditions which were specified in Tables 2 to Table 9. We also calculate delays when a collaboration is nested within another collaboration. Even though this is not mentioned in Table 21, we test this functionality as well.

Table 21 – Test cases for Sequence Operators

Strict Sequence:

Single Control Flow	Fixed	Range	Stochastic Distribution
Delay	1: X 2: X 3: X 4: X	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X
Path	1: X 2: X 3: X 4: X	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X

Multiple Control Flow	Range	Stochastic Distribution
Delay	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X
Path	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X

Weak Sequence:

Single Control Flow	Fixed	Range	Stochastic Distribution
Delay	1: X 2: X 3: X 4: X	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X
Path	1: X 2: X 3: X 4: X	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X

Multiple Control Flow	Range	Stochastic Distribution
Delay	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X
Path	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X

Concurrency:

Single Control Flow	Fixed	Range	Stochastic Distribution
Delay	1: X 2: X 3: X 4: X	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X
Path	1: X 2: X 3: X 4: X	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X

Multiple Control Flow	Range	Stochastic Distribution
Delay	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X
Path	max: 1: X 2: X min: 1: X 2: X	1: X 2: X 3: X 4: X

Alternative:

Single Control Flow	Fixed	Range	Stochastic Distribution
Delay	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X 4a: X 4b: X 5a: X 5b: X	max: 1: X 2: X 3: X 4: X 5: X min: 1: X 2: X 3: X 4: X 5: X	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X 4a: X 4b: X 5a: X 5b: X
Path	1a: X 1b: X 2a: X 2b: X	max: 1: X 2: X 3: X 4: X 5: X	1a: X 1b: X 2a: X 2b: X

	3a: X 3b: X 4a: X 4b: X 5a: X 5b: X		3a: X 3b: X 4a: X 4b: X 5a: X 5b: X
		min: 1: X 2: X 3: X 4: X 5: X	

Multiple Control Flow	Range of Delays (Max)	Range of Delays (Min)
Delay	1: X 2: X 3: X 4: X 5: X 6: X 7: X 8: X 9: X	1: X 2: X 3: X 4: X 5: X 6: X 7: X 8: X 9: X
Path	1: X 2: X 3: X 4: X 5: X 6: X 7: X 8: X 9: X	1: X 2: X 3: X 4: X 5: X 6: X 7: X 8: X 9: X

Multiple Control Flow	Stochastic Delays
Delay	1: X 2: X 3: X 4: X 5: X 6: X 7: X 8: X 9: X
Path	1: X 2: X 3: X 4: X 5: X 6: X 7: X 8: X 9: X

Strict While Loop:

Single Control Flow	Fixed	Range	Stochastic Distribution
Delay	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X	max: 1a: X 2a: X max: 1b: X 2b: X min: 1a: X 2a: X min: 1b: X 2b: X	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X
Path	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X	max: 1a: X 2a: X max: 1b: X 2b: X max: 1a: X 2a: X max: 1b: X 2b: X	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X

Multiple Control Flow	Range	Stochastic Distribution
Delay	max: 1: X min: 2a: X 2b: X	1a: X 1b: X
Path	min: 1: X min: 2a: X 2b: X	1a: X 1b: X

Weak While Loop:

Single Control Flow	Fixed	Range	Stochastic Distribution
Delay	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X	max: 1a: X 2a: X max: 1b: X 2b: X min: 1a: X 2a: X min: 1b: X 2b: X	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X
Path	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X	max: 1a: X 2a: X max: 1b: X 2b: X max: 1a: X 2a: X max: 1b: X 2b: X	1a: X 1b: X 2a: X 2b: X 3a: X 3b: X

Multiple Control Flow	Range	Stochastic Distribution
Delay	max: 1: X	1a: X 1b: X
	min: 2a: X 2b: X	
Path	min: 1: X	1a: X 1b: X
	min: 2a: X 2b: X	

ANNEX C – NORMAL DISTRBUTION APPROXIMATION

C.1 Analysis of X1~N(0,1), X2~N(4,1)

We examine in Table 22 the PDF and CDF of a normal distribution variable X1 with mean of 0 and standard deviation of 1 and of a normal distribution variable X2 with mean of 4 and standard deviation of 1. We use Equation (3) and (4) to calculate the minimum and the maximum delay. With the values in the table, we calculate the mean of the minimum and the maximum by determining when the probability for minimum and maximum graph is 0.5 (the mean of a normal distribution is found when the probability is 0.5).

The minimum distribution is identical to X1 and the mean is at 0. The maximum distribution is identical to X2 and mean is at 3.75. This conforms to what we proposed in Chapter 6, namely that if there is *little overlap*, then the minimum of the two distributions will be the distribution with smaller mean and vice versa for the maximum.

Table 22 – PDF and CDF of X1~N(0,1), X2~N(4,1)

	PDF		CDF			
	X1	**X2**	**X1**	**X2**	**max(X1,X2)**	**min(X1,X2)**
-4	0.00013383	5.05227E-15	3.16712E-05	6.22096E-16	1.97026E-20	3.16712E-05
-3.75	0.000352596	3.61829E-14	8.84173E-05	4.59463E-15	4.06244E-19	8.84173E-05
-3.5	0.000872683	2.43432E-13	0.000232629	3.19089E-14	7.42294E-18	0.000232629
-3.25	0.002029048	1.53854E-12	0.000577025	2.08386E-13	1.20244E-16	0.000577025
-3	0.004431848	9.13472E-12	0.001349898	1.27981E-12	1.72762E-15	0.001349898
-2.75	0.009093563	5.09494E-11	0.002979763	7.39226E-12	2.20272E-14	0.002979763
-2.5	0.0175283	2.66956E-10	0.006209665	4.016E-11	2.4938E-13	0.006209665
-2.25	0.031739652	1.314E-09	0.012224473	2.05226E-10	2.50878E-12	0.012224473
-2	0.053990967	6.07588E-09	0.022750132	9.86588E-10	2.2445E-11	0.022750133

-1.75	0.086277319	2.63924E-08	0.040059157	4.46217E-09	1.78751E-10	0.040059161
-1.5	0.129517596	1.07698E-07	0.066807201	1.89896E-08	1.26864E-09	0.066807219
-1.25	0.182649085	4.12847E-07	0.105649774	7.60496E-08	8.03462E-09	0.105649842
-1	0.241970725	1.48672E-06	0.158655254	2.86652E-07	4.54788E-08	0.158655495
-0.75	0.301137432	5.02951E-06	0.226627352	1.01708E-06	2.30499E-07	0.226628139
-0.5	0.352065327	1.59837E-05	0.308537539	3.39767E-06	1.04831E-06	0.308539888
-0.25	0.386668117	4.77186E-05	0.401293674	1.06885E-05	4.28924E-06	0.401300074
0	0.39894228	0.00013383	**0.5**	3.16712E-05	1.58356E-05	**0.500015836**
0.25	0.386668117	0.000352596	0.598706326	8.84173E-05	5.2936E-05	0.598741807
0.5	0.352065327	0.000872683	0.691462461	0.000232629	0.000160854	0.691534236
0.65	0.301137432	0.002029048	0.773372648	0.000577025	0.000446255	0.773503417
0.75	0.241970725	0.004431848	0.841344746	0.001349898	0.00113573	0.841558914
1	0.182649085	0.009093563	0.894350226	0.002979763	0.002664952	0.894665038
1.25	0.129517596	0.0175283	0.933192799	0.006209665	0.005794815	0.933607649
1.5	0.086277319	0.031739652	0.959940843	0.012224473	0.011734771	0.960430545
1.75	0.053990967	0.053990967	0.977249868	0.022750132	0.022232563	0.977767437
2	0.031739652	0.086277319	0.987775527	0.040059157	0.039569455	0.988265229
2.25	0.0175283	0.129517596	0.993790335	0.066807201	0.066392351	0.994205185
2.5	0.009093563	0.182649085	0.997020237	0.105649774	0.105334962	0.997335048
2.75	0.004431848	0.241970725	0.998650102	0.158655254	0.158441086	0.99886427
3	0.002029048	0.301137432	0.999422975	0.226627352	0.226496583	0.999553745
3.25	0.000872683	0.352065327	0.999767371	0.308537539	0.308465764	0.999839146
3.5	0.000352596	0.386668117	0.999911583	0.401293674	0.401258193	0.999947064
3.75	0.00013383	0.39894228	0.999968329	**0.5**	**0.499984164**	0.999984164
4	4.77186E-05	0.386668117	0.999989311	0.598706326	0.598699926	0.999995711
4.25	1.59837E-05	0.352065327	0.999996602	0.691462461	0.691460112	0.999998952
4.5	5.02951E-06	0.301137432	0.999998983	0.773372648	0.773371861	0.99999977
4.75	1.48672E-06	0.241970725	0.999999713	0.841344746	0.841344505	0.999999955
5	4.12847E-07	0.182649085	0.999999924	0.894350226	0.894350158	0.999999992
5.25	1.07698E-07	0.129517596	0.999999981	0.933192799	0.933192781	0.999999999
5.5	2.63924E-08	0.086277319	0.999999996	0.959940843	0.959940839	1
5.75	6.07588E-09	0.053990967	0.999999999	0.977249868	0.977249867	1
6	1.314E-09	0.031739652	1	0.987775527	0.987775527	1
6.25	2.66956E-10	0.0175283	1	0.993790335	0.993790335	1
6.5	5.09494E-11	0.009093563	1	0.997020237	0.997020237	1
6.75	9.13472E-12	0.004431848	1	0.998650102	0.998650102	1
7	1.53854E-12	0.002029048	1	0.999422975	0.999422975	1
7.25	2.43432E-13	0.000872683	1	0.999767371	0.999767371	1

7.5	3.61829E-14	0.000352596	1	0.999911583	0.999911583	1
7.75	5.05227E-15	0.00013383	1	0.999968329	0.999968329	1
8	0.00013383	5.05227E-15	3.16712E-05	6.22096E-16	1.97026E-20	3.16712E-05

C.2 Analysis of X_1~N(1.5,1.5), X_2~N(1,1)

We examine in Table 23 the PDF and CDF of a normal distribution variable X1 with mean of 1.5 and standard deviation of 1.5 and of a normal distribution variable X2 with mean of 1 and standard deviation of 1. We use Equation (3) and (4) to calculate the minimum and the maximum delay. With the values in the table, we calculate the mean of the minimum and the maximum by determining when the probability for minimum and maximum graph is 0.5 (the mean of a normal distribution is found when the probability is 0.5).

The mean of the minimum and maximum is very close to the value approximated by Equations (94) and (95).

Table 23 – PDF and CDF of X_1~N(1.5,1.5), X_2~N(1,1)

	PDF		**CDF**			
	X1	**X2**	**X1**	**X2**	**max(X1,X2)**	**min(X1,X2)**
-4	0.00032018	1.48672E-06	0.000122866	2.86652E-07	3.52198E-11	0.000123153
-3.75	0.000581788	5.02951E-06	0.000232629	1.01708E-06	2.36603E-10	0.000233646
-3.5	0.001028186	1.59837E-05	0.00042906	3.39767E-06	1.45781E-09	0.000432457
-3.25	0.001767317	4.77186E-05	0.000770985	1.06885E-05	8.24069E-09	0.000781665
-3	0.002954566	0.00013383	0.001349898	3.16712E-05	4.27529E-08	0.001381527
-2.75	0.004804067	0.000352596	0.002303266	8.84173E-05	2.03649E-07	0.00239148
-2.5	0.007597324	0.000872683	0.003830381	0.000232629	8.91058E-07	0.004062119
-2.25	0.011685534	0.002029048	0.006209665	0.000577025	3.58313E-06	0.006783107
-2	0.017481259	0.004431848	0.009815329	0.001349898	1.32497E-05	0.011151977
-1.75	0.025435082	0.009093563	0.01513014	0.002979763	4.50842E-05	0.018064819
-1.5	0.035993978	0.0175283	0.022750132	0.006209665	0.000141271	0.028818527
-1.25	0.049540776	0.031739652	0.033376508	0.012224473	0.00040801	0.04519297

-1	0.066318093	0.053990967	0.047790352	0.022750132	0.001087237	0.069453247
-0.75	0.086345064	0.086277319	0.066807201	0.040059157	0.00267624	0.104190118
-0.5	0.10934005	0.129517596	0.09121122	0.066807201	0.006093566	0.151924855
-0.25	0.13466579	0.182649085	0.121672505	0.105649774	0.012854673	0.214467606
0	0.161313816	0.241970725	0.158655254	0.158655254	0.02517149	0.292139018
0.25	0.18794125	0.301137432	0.202328381	0.226627352	0.045853145	0.383102588
0.5	0.212965337	0.352065327	0.252492538	0.308537539	0.077903426	**0.48312665**
0.65	0.226511583	0.375240347	0.285470336	0.363169349	0.103674076	**0.544965609**
0.75	0.234710218	0.386668117	0.308537539	0.401293674	0.123814163	0.58601705
1	0.251588818	0.39894228	0.36944134	0.5	0.18472067	0.68472067
1.25	0.262293144	0.386668117	0.433816167	0.598706326	0.259728484	0.772794009
1.5	0.26596152	0.352065327	0.5	0.691462461	0.345731231	0.845731231
1.75	0.262293144	0.301137432	0.566183833	0.773372648	0.43787109	0.901685391
2	0.251588818	0.241970725	0.63055866	0.841344746	0.530517216	0.94138619
2.25	0.234710218	0.182649085	0.691462461	0.894350226	0.618409609	0.967403079
2.5	0.212965337	0.129517596	0.747507462	0.933192799	0.697568581	0.98313168
2.75	0.18794125	0.086277319	0.797671619	0.959940843	0.765717567	0.991894896
3	0.161313816	0.053990967	0.841344746	0.977249868	0.822204042	0.996390572
3.25	0.13466579	0.031739652	0.878327495	0.987775527	0.867590405	0.998512618
3.5	0.10934005	0.0175283	0.90878878	0.993790335	0.903145506	0.999433609
3.75	0.086345064	0.009093563	0.933192799	0.997020237	0.930412105	0.99980093
4	0.066318093	0.004431848	0.952209648	0.998650102	0.950924262	0.999935488
4.25	0.049540776	0.002029048	0.966623492	0.999422975	0.966065726	0.999980741
4.5	0.035993978	0.000872683	0.977249868	0.999767371	0.977022531	0.999994708
4.75	0.025435082	0.000352596	0.98486986	0.999911583	0.98478278	0.999998662
5	0.017481259	0.00013383	0.990184671	0.999968329	0.990153311	0.999999689
5.25	0.011685534	4.77186E-05	0.993790335	0.999989311	0.993779713	0.999999934
5.5	0.007597324	1.59837E-05	0.996169619	0.999996602	0.996166235	0.999999987
5.75	0.004804067	5.02951E-06	0.997696734	0.999998983	0.997695719	0.999999998
6	0.002954566	1.48672E-06	0.998650102	0.999999713	0.998649816	1