Schedule Quality Assessment for 4D Models using Industry Foundation Classes

Ashok Kavad (40037503)

Rahul Dharsandia (40042642)

Advisor: Mazdak Nik-Bakht

Report

ENGR 6971 Project and Report I

Department of

Building, Civil and Environmental Engineering

Submitted for Partial Fulfillment of Requirement for the Degree of

Master of Engineering in Civil Engineering

Concordia University

Montreal, Quebec, Canada

December 2018

REPORT CONTRIBUTION TABLE

Ashok Kavad	**Rahul Dharsandia**
Acknowledgement	Abstract
Chapter 1	Chapter 3
Chapter 2	Chapter 4
Chapter 6	Chapter 5
Appendix A	Appendix B

ABSTRACT

Schedule assessment models are developed to ensure the proper development of a schedule. Most of the existing automated models are targeted towards two-dimensional schedules, and not four-dimensional, despite the emergence of building information modeling in the construction industry. This study presents the adaptation of the existing schedule quality assessment criteria to evaluate two-dimensional models onto four-dimensional models, utilizing building information modeling and Industry Foundation Classes (IFC). The study starts with a comprehensive review of previous schedule assessment models and identifying the major checks performed in literatures and industry schedule review tools. The checks are then categorized as quantifiable and qualitative, to differentiate between the measures that can be fully automated and others which would require expert intervention. Afterwards, the study presents the methodology for attaining the inputs required for the quantitative measures in 4D models. The methodology revolves around using Industry Foundation Classes (IFC), as a standard data model for storing building and construction data. Accordingly, a technological review was conducted of the existing 4D modeling software, to view the capabilities and limitations that could affect the development of a schedule assessment model. Initial algorithms were developed to measure the wellness of schedule. These developed algorithms were then validated and verified, by testing them versus different schedules with known errors.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our advisor, professor Mazdak Nik-Bakht for providing their invaluable guidance, comments and suggestions throughout the course of the research. We would like to offer our sincere appreciation for providing learning opportunities through this research work.

We would also like to thank Ph. D student Abdelhady Hosny for his continuous support in the research. We are thankful to him for providing his time to solve the problems during the research.

It was impossible to deliver the content of this research work clearly without the help of professor Mazdak Nik-Bakht and Ph.D. student Abdelhady Hosny. Thank you again for providing your valuable time and support.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 General

Building Information Modeling (BIM) is not a new technology, BIM has been in development since late 70s. However, BIM is now only gaining momentum in Canadian construction industry. According to International BIM Report 2017 by National BIM Standard, 78% Canadians think that BIM is the future for managing project information and 67% are currently using BIM in their organization. Building Information Modeling is the process of collaboratively developing and managing an integrated digital model containing geometry data and lifecycle information of propose or existing facility [1]. The design and construction industry are now transitioning from use of two-dimensional CAD and paper for design to three-dimensional digital models loaded with information. The BIM usage in project is expected to continue growing sharply in coming years [2].

The Canadian construction industry contributes more than 12% of Canada’s gross domestic product (GDP). The employees working in construction industry account for 6% of Canadian employment [3]. The construction projects are complex and requires planning and management from early design phase to construction phase. The planning involves forecasting the future activities and outcomes that are uncertain. The schedule preparation is essential part of the project planning, it determines logical sequence of the construction activities with the calendar dates for starting and finishing of the activities. The schedule provides overview of how and when the project milestones to be delivered as defined in the project scope [4]. These construction schedules are often prepared by the contractors once they are awarded and required to submit to the owner or the agent of the owner for review. The detailed schedule is used for project execution, tracking the project status, reporting the progress and during the construction claims and disputes [5].

Several factors are to be consider for smooth running of the project, quality of the schedule is one of the factors require serious attention. The quality schedule helps in improving overall productivity and quality of construction production, if the quality schedule is loaded with cost and other technical information, the project stakeholders can use it for project management decision making. Thus, good quality schedule is vital in selection of the construction strategy and identifying the problems. According to Government Accountability Office (GAO), quality of the scheduling practice and quality of the schedule itself play an important role in final success of the construction projects [4]. The previous research says that project started with well developed schedule in early phase will outperform in terms of project time and cost performance [3].

The development in Computer Aided Design (CAD) technologies have allowed to link the time schedule with the three-dimensional model to create 4D model. The 4D model allow planner to visualize the how the construction is going to progress and help in identifying the potential problems like missing activities, scheduling conflicts, etc. 4D visualization of construction plan allows planner to modify the production process prior to construction if issues identified in activity sequencing. The 4D visualization can also be used to compare actual progress against baseline of the project. Thus, the 4D visualisation technique provides an effective solution to communicate temporal and spatial information to the project team [6].

1.2 Industry Foundation Classes (IFC)

A typical project requires collaboration and information exchange between different disciplines involved in project development. Traditionally, information exchange was carried out using paper drawings and documents. The construction industry is moving towards the use of BIM due to technological development and the digital design models are being used as a medium of information exchange. However, different disciplines use various software applications from different vendors that information exported from one software might not be compatible with the other software. These compatibility issue poses challenge for the model exchange [2].

Industry Foundation Classes (IFC) defined by the BuildingSMART are the open and neutral data format allow the sharing of information between all the parties, regardless of which software application they are using for design, construction, procurement, maintenance and operations. The IFC is one of five types of open standard defined by BuildingSMART, each of which uses to perform different functions in the information delivery. The digital model in IFC data format can hold and exchange relevant data between different software applications [7]. A model exported in IFC format from any design authoring software consists not only three-dimensional geometry related data but also the metadata of that geometry. For example, if we consider door, IFC enables to store its geometrical data like shape and size, and informational data or metadata like costing, thermal performance, fire safety performance, etc. Many of the software applications or BIM tools currently used in AEC industry support import and export of IFC files. Furthermore, this IFC file could also be used for design analysis, clash detection, cost estimation and construction sequencing [2]. IFC was first defined in 1996 by the International Alliance for Interoperability (IAI), now known as BuildingSMART and since then there has been number of minor and major revisions.

The following figure shows summary of IFC releases timeline. IFC4 is current version of data standard [8].

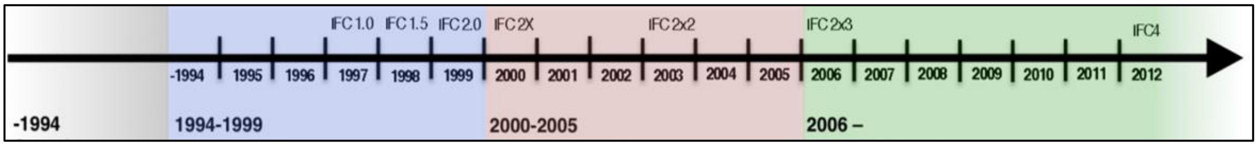

Figure 1: IFC Timeline *[8]*

From a technical point of view, IFC is defined using the ISO 10303 suite of specifications for data modelling and exchange, also known as STEP (Standard for the Exchange of Product Data). STEP consists of a range of specifications, a language for specifying data schema (STEP/Express, in which the IFC language is defined), a mapping (Part-21) for text file representation of models conforming to that schema, a mapping (StepXML) for XML file representation of models, and mappings to Application Programming Interface (API) for accessing models programmatically (Part-22, Standard Data Access Interface, or SDAI). The Part-21 mapping defines the IFC file format [2].

1.3 Motivation

The information like construction drawings, specifications and schedule required during planning are generally in different forms which create difficulties in integrating all the information manually. This motivated us to use building information modeling as it facilitates integrating all the information in one place digitally and can be accessed easily. Further, despite the emergence of building information modeling in the construction industry, existing automated schedule assessment models in previous research and industry tools review only two-dimensional schedules, and not four-dimensional. The two-dimensional (2D) schedule refers to schedule developed in scheduling tools like MS Project which has only temporal information of construction activities. The four-dimensional (4D) schedule refers to schedule information available in 4D model which has both temporal and spatial information. So far, no research has been conducted to assess quality of 4D schedule in IFC format. This motivated us to develop new schedule assessment model to review 4D schedule in IFC format using schedule quality check measurements done at 2D level in previous literatures.

1.4 Problem Statement

IFC is open format used to share information for design analysis, simulation and management in the BIM project. IFC file format can store the temporal (i.e. schedule) and spatial information of the 4D model. The idea of this research is to assess quality of the 4D schedule in IFC format instead of reviewing 2D schedule. In the BIM project, contractor has to submit 4D model showing how the project is going to progress, then the owner or his agent has to review the planned progress for whether it is meeting the stakeholders' expectation or not. However, for complex projects, just looking at the 4D simulation of the planned project progress, it is difficult to determine whether this planned 4D schedule is realistic or not. Furthermore, currently available commercial tools are not able to assess the quality of 4D schedule in IFC format. Thus, there is need of tool that can be used to evaluate 4D schedule in IFC format and to do so the tool should be able to access schedule data within the 4D IFC environment. Here, 4D IFC is referred to IFC file of 4D model exported from software application like Synchro. The planner can have IFC file of 4D model and assess quality of the 4D schedule in IFC format using proposed automated schedule assessment model.

1.5 Objective

The main objective of the study is to develop the methodology to assess quality of the schedule in the 4D IFC environment. The following are the set of sub-objectives defined in order to achieve the primary objective.

a) To conduct comprehensive review of previous schedule assessment models to identify the major schedule checks performed.

b) To review the various literatures and industry schedule review tools to identify the criteria, measures and metrics used to assess the schedule quality.

c) To categorize metrics into quantitative and qualitative to differentiate between the measures that can be fully automated and others which would require expert intervention.

d) To identify a 4D modeling software that has IFC working capabilities, through which we would export IFC files that can be evaluated in proposed schedule assessment model.

e) To develop schedule quality assessment model (algorithms) for 4D models that uses IFC.

f) To validate and verify developed algorithms by testing them versus different schedules with known errors.

1.6 Scope of report

The review of schedule quality assessment models from literature and industry schedule analysis tools are presented in Chapter 2. It also highlights the criteria used to evaluate the schedule quality in literatures and industry tools. Chapter 3 presents propose methodology of the research work starting with developing conceptual framework, classification of schedule check metrics, receiving 4D IFC, procedure of developing schedule quality assessment model or algorithms to assess the schedule quality. After that developed model is validated by testing the algorithms against different scenarios with known errors in schedule, the procedure to validate model is described in Chapter 4. The Chapter 5 consists the limitation of IFC and software applications. The conclusions of study are given in the Chapter 6. The recommendation for the future works is also presented in this chapter.

2 LITERATURE REVIEW

2.1 General

Moosavi and Moselhi (2014) introduced Schedule Development Index (SDI) to check fitness of the schedule. They conducted online questionnaire survey and reviewed various literatures and found the important criteria use to evaluate the schedule. They grouped the criteria into obligatory criteria and complementary criteria. Schedule must satisfy the obligatory criteria if not schedule must be rejected. Once schedule satisfies all obligatory criteria, schedule is assessed for complementary criteria and SDI is calculated. Schedule with SDI $\geq$ 800 is excellent schedule, 800 > SDI $\geq$ 500 is good schedule and SDI < 500 is acceptable schedule. They developed automated Schedule Assessment and Evaluation (SAE) software application to assess the detailed construction schedule for job logic, productivity and crew size. SAE reads the schedule and recognizes the predefine keywords of activity (e.g. keyword rebar means installation of rebars) and check for typical predecessors and successors (e.g. pouring activity must follow concreting work) incorporated in application. If schedule does not follow the typical relationship it will highlight activities in the schedule and creates a report of activities. SAE also evaluate the schedule for productivity and crew size using own database extracted from RS Means, if planned productivity and crew size deviate 30% from RS Means data, activities are highlighted in schedule and creates a report of same. They presented set of empirical rules to review job logic based on historical data of recently completed projects.

Table 1: Empirical rules for starting of different trades *[5]*

No.	Empirical Rules
1	Duration of foundation trade is approximately 5% of framing trade duration
2	When more than 30% of foundation is complete, framing trade can start
3	Duration of framing trade is approximately 35% of project duration
4	Once framing of three floors is performed, curtain wall trade could start
5	Duration of curtain wall trade is approximately 30% of project duration
6	Once 30% of curtain wall is complete, architectural trade starts
7	Duration of architectural trade is approximately 40% of project duration
8	HVAC and electrical trades could start at the same time once 30% of framing is complete
9	Duration of electrical trade is approximately 60% of project duration
10	Duration of HVAC trade is approximately 65% of project duration
11	Once 10% of HVAC is complete, firefighting trade starts
12	Duration of firefighting trade is approximately 30% of project duration
13	Once framing is complete, elevator and escalator trade starts
14	Duration of elevator and escalator trade is approximately 30% of project duration

US Defence Contract Management Agency – DCMA (2012) introduced 14 metrics to assess the schedule health. DCMA 14 metrics are: Logic, Leads, Lags, Relationship Types, Hard Constraints, High Float, Negative Float, High Duration, Invalid Dates, Resources, Missed Tasks, Critical Path, Critical Path Length Index and Baseline Execution Index. This guide can be used to review initial schedule and in progress schedule. DCMA defined threshold value for each metrics. The number of tasks without predecessors and/or successors should not be exceed 5%. There should not be any activities with lead. The number of tasks with lags should not be more than 5%. Schedule should have at least 90% finish to start relationships. The number of activities with hard constraints should be maximum 5%. The percentage of tasks with float more than 44 days should not exceed 5%. DCMA does not allow any negative float in schedule. Number of tasks with duration more than 44 days should be limited to 5%. Tasks should not have invalid dates (e.g. tasks must not have actual start and/or finish date in future). Each task should be assigned resources in the schedule. The number of tasks with missing baseline finish date, in other words number of tasks which has actual finish date later than the planned finish date, should not exceed 5%. All the critical activities must be logically linked, there should not be any missing logic. The Critical Path Length Index (CPLI) is a measure of the efficiency required to complete the milestone on time when constraint is assigned to milestone completion, it should not be less than 0.95 value. Baseline Execution Index (BEI) is measure of the efficiency at which tasks have been performed when measured against the baseline. When BEI is less then 1, then project is moving behind the schedule. BEI should not be less than 0.95 value.

Bragadin and Kahkonen (2016) did research on improving Quality of schedules oriented towards Construction. Their findings are based on a literature study from variety of papers. These findings combined with the practical experiences were used to define various metric for measuring quality of schedule. They included 75 schedule requirements which they classified into 5 groups, i.e., general requirements, construction process, schedule mechanics, cost and resources and control process. These groups are known as schedule health indicators and have sub groups too. Based on every project, these indicators have different relative weights and further down sub-groups would have its own relative weight. Then, simple weighing formulas were used where multiplication of points in each subgroup with relative weight would take place and finally the quality percent of project would be calculated. The best part is that project scheduler/manager can set relative weights according to his priorities. Even, performance graphs are extracted to have broad idea that which group is lagging most in quality. The structure created here is used to assess quality of construction schedule which they termed as "Schedule Health Assessment." The developed method was intended to help project planners for producing and maintaining good quality schedules right from the very beginning of project through construction phase till its end. The best part of paper is that it introduced a checklist of detailed requirements which behaves itself as a guide.

Bansal and Pal (2008, 2009) used Geographic Information System (GIS) based 3D animation to review construction schedule for job logic, complexity and completeness. They generated 3D model and schedule on single platform using ArcGIS. They created 4D model by linking activities of schedule with corresponding 3D elements. The main objective of this paper was to develop and review the 4D model on single platform like GIS. However, proposed methodology can also be used for 3D models generated in another CAD software and schedule developed in MS Project or Primavera. The proposed approach is not automated, user has to run animation and look for any discrepancies related to job logic, complexity and completeness of the schedule. User uses the 4D visualisation to review schedule for job logic by checking missing activity relationships and incorrect construction sequence. For example, if small portion of brickwork above the entrance appears before the support for that brickwork portion, this construction sequence is to be corrected by the user. Complexity of schedule is measured by level of detail for activity given in the schedule. Completeness of the schedule is evaluated by missing activities using in-house scripts written in Avenue (Programming language in ArcView). This script informs the user about the activities which are not assigned to any elements of 3D model.

Krzeminski (2016) developed schedule equalization model to derive alternative schedules from base schedule. This model utilizes efficiency coefficient of brigade working on tasks to derive alternative schedules. Krzeminski also introduced objective functions. Once alternate schedules are derived from the base schedule using schedule equalization model, all schedules are evaluated using objective functions and optimal schedule is determined. The optimal schedule is one where reduction in brigade downtime is achieved without prolonging the brigades' working time unduly. In the objective functions, the base schedule and alternate schedules are compared to check how well alternate schedule is doing over base schedule in terms of decrease in project duration, reduction in brigade downtime and increase in labor input. The objective functions are calculated and schedule with the highest value of objective functions is considered the optimal schedule.

Bragadin and Kahkonen (2015) presented 3 "S" rule of construction process to assess quality of the schedule. 3 "S" are Safety, Space and Structure meaning that schedule must be prepared considering safe working environment for labors, enough space should be available to avoid any time space conflict on site and required sequence of construction process and project phase. The aim of the study was to implement a schedule quality assessment method that considers the 3 "S" rule. It is challenging tasks to load the schedule with the space related information. They presented solution to address time space conflict by integrating two scheduling tools CPM method and flow line or linear scheduling method. The project manager can use this solution to modify production models by changing construction methods, sequences to minimize the time space conflict.

Garcia de Soto, et al. (2017) proposed the methodology to determine optimal schedule and then review it by integrating project visualization. In the first step of methodology Tabu search algorithm is used to determine optimal schedule considering project duration, costs and resources. First initial schedule is developed using CPM method in combination with PERT using simulation of 1000 iteration. The duration of activities is assumed to have beta distribution. The Tabu search algorithm is applied to developed different scenarios and objective function is calculated for each scenario. The scenarios are developed using different combination of weight (0 to 2) and factors, i.e., project duration, costs and resource. The scenario with least value of the objective function is considered optimal schedule. In the second step 4D model is generated, the visualization of the optimized schedule allows the project team to check manually the schedule for missing activities and ensure that sequencing and constructability requirements are satisfied.

Bansal (2011) utilized Geographic Information System (GIS) for space planning. Set of validation rules for working areas, access path and locations were presented to identify time space conflict. GIS was used to generate multiple types of spaces, i.e., cubical, spherical, prism corresponding to various activities. GIS base area topology was implemented through a set of validation rules that define how working areas have to share the jobsite. In the proposed methodology first space data would be entered then topography is validated for rules violation when validation rules are violated, it generates error about the time space conflict and reports it to planner and provides opportunity to resolve the serious errors. Thus, it enabled time space conflict resolution prior to the construction. It would be difficult to enter space related data during the planning of the project schedule as hundreds of activities are involved, so this server base GIS technology facilitates the project engineers to add space related information required for the activities on site. Thus, space could be assigned and checked for conflict prior to execution of the activities in construction phase. The validation rules are illustrated in the following table.

Table 2: Validation rules for area, path and location *[9]*

Description	**Validation Rules**
Validation rules for Working Area	Areas corresponding to one activity must not overlap with areas corresponding to another activity
	Must not have gaps within a single area or between adjacent areas, all areas must form continuous surface
	An activity must share all its areas with the areas corresponding to another activity
	Each area corresponding to an activity must contain at least one location
Validation rules for Access Path	Path segments do not overlap with another path segment at one time
	Paths must be covered by the boundaries of area features
	Endpoints of paths must be covered by a few specific locations on the jobsite
Validation rules for Location	Locations must fall within the working areas
	Locations must be covered by the endpoints of paths
	Locations must be covered by paths

Marx and Konig (2013) modeled spatial requirements, i.e., bounding boxes of activities and evaluated schedule for spatial conflict using building information models. Spatial Constraint Query Language (SCQL) extensions of Partial Model Query Language (PMQL) used for IFC was developed to define spatial requirements efficiently. The spatial requirements integrated into a constraint-based simulation approach to identify and solve spatial conflicts. The constraint-based simulation means that each time an event occurs, the constraints are checked to identify which processes can be started next. During the simulation run, collision between bounding boxes was calculated, hierarchical axis aligned bounding boxes algorithm was used to calculate the collision or intersection. If spatial conflicts are detected, activities were rescheduled accordingly by increasing activity durations or postponing start dates.

Kassem, et al. (2015) presented methodology to manage workspace within the Industry Foundation Class (IFC) compliant 4D tool. They created 4D model by linking schedule in XML format with 3D model in IFC format. Then model was explored in game engine environment using #.Net programming language. This methodology facilitates generation and allocation of workspace during the 4D simulation and detects time space conflict through intersection test. The workspaces are graphically represented as axis aligned bounding boxes (AABBs). The intersection test of bounding boxes uses clash rules to detect the conflicts between workspaces. In the test, coordinates of two bounding boxes are compared in each cartesian direction, if clash rules are satisfied for two bounding boxes, a conflict is detected. These results of workspace conflict are then stored in central database and can be used for further decision making.

There are many industry applications available to review the construction schedule, i.e., Project analyzer, Schedule inspector, Schedule analyzer, Acumen fuse, Open plan, Schedule cracker, P6 schedule checker, XER reader, XER schedule toolkit. All the applications evaluate the schedule for the DCMA 14 metrics and some of the applications also review the schedule for the metrics other than the DCMA 14 metrics. These applications have brought automation in schedule evaluation process, so planner does not require to analyse the schedule manually. These desktop base application reviews only 2D time schedule prepared in MS Project or Primavera, there is not such tool available that can retrieve the schedule related data from 4D IFC environment and review the schedule. The following tables summarize the criteria used to evaluate performance of the construction schedule in various literatures and industry software.

Table 3: Criteria reviewed in literatures

Criteria / References	Bansal and Pal (2008)	Bansal and Pal (2009)	DCMA (2012)	Moosavi and Moselhi (2014)	Bragadin and Kahkonen (2015)	Krzeminski (2016)	Bragadin and Kahkonen (2016)	Garcia de Soto, et al. (2017)	Bansal (2011)	Marx and Konig (2013)	Kassem, et al. (2015)
Network and logic	x	x	x	x	x		x	x			
Resources			x	x		x	x	x			
Critical path			x	x			x				
Float			x	x			x				
Lag & Lead (negative lag)			x	x			x				
Soft & hard Constraints			x	x			x				
Invalid dates and missed tasks			x				x				
Schedule baseline			x				x				
Schedule definition/overview				x		x	x	x			
Workspace				x	x		x		x	x	x
Activity definition				x			x				
Activity duration				x			x				
Monetary value/cost of activities							x	x			
Completeness of schedule	x	x						x			
Project total level of effort				x		x	x				
Activity progress evaluation							x				
Activity timing				x			x				

Complexity of schedule	x	x									
Construction process productivity					x		x				
Project cost ratio				x			x				
Schedule process				x			x				
Activity mis-assignments							x				
Activity sequencing							x				
Buffers (Reserves)							x				
Schedule projections (Conformance)							x				
Schedule scope				x							
Special considerations				x							

Table 4: Criteria reviewed in industry software

Criteria / References	Project Analyzer	Schedule Inspector	Schedule Analyzer	Acumen Fuse	Open Plan	Schedule Cracker	P6 Schedule Checker	XER Reader	XER Schedule Toolkit
Network and logic	x	x	x	x	x	x	x	x	x
Resources	x	x	x	x	x	x	x	x	x
Critical path	x	x	x	x	x	x	x	x	x
Float	x	x	x	x	x	x	x	x	x
Lag & Lead (negative lag)	x	x	x	x	x	x	x	x	x
Soft & hard Constraints	x	x	x	x	x	x	x	x	x
Invalid dates and missed tasks	x	x	x	x	x	x	x	x	x
Schedule baseline	x	x	x	x	x	x	x	x	x
Schedule definition/overview	x	x	x						
Workspace									
Activity definition	x	x	x						
Activity duration	x		x						
Monetary value/cost of activities			x						x
Completeness of schedule									
Project total level of effort									
Activity progress evaluation									x
Activity timing									
Complexity of schedule									
Construction process productivity									

Project cost ratio									
Schedule process									
Activity mis-assignments									
Activity sequencing									
Buffers (Reserves)									
Schedule projections (Conformance)									
Schedule scope									
Special considerations									

3 METHODOLOGY

3.1 Conceptual framework

The methodology starts with developing conceptual framework (Appendix A) of schedule quality evaluation criteria, measures and metrics from various literature and industry schedule review tools. All the schedule check metrics found were categorized into quantitative, qualitative and rule of thumb metrics. The quantitative metrics are one that can be fully automated, the qualitative metrics require expert intervention, the metrics which are difficult to determined between quantitative and qualitative, are classified as rule of thumb metrics. The metrics which are subjective or require extra information, were categorized into unclear metrics. Further, the quantitative metrics are divided into planning and control related metrics. The developed methodology focuses only on the planning related quantitative metrics. The metrics come under the planning are further divided into bimable metrics and non-bimable metrics. The bimable metrics are the metrics which digital representation is possible in BIM. The effort was made to prepare algorithms or schedule assessment model of bimable metrics but due to limitation of IFC and software application, it was not possible to prepare model for all bimable metrics. The limitations of IFC and software application are described in Chapter 5. The metrics for which algorithms are possible to prepare were classified as programmable metrics and others were classified as not programmable metrics. The classification of schedule check metrics is given in Figure 2. The numbers given in the parenthesis shows number of metrics within classification. Total 168 schedule check metrics were identified through review of various literature and industry schedule review tools and algorithms prepared consider 51 metrics.

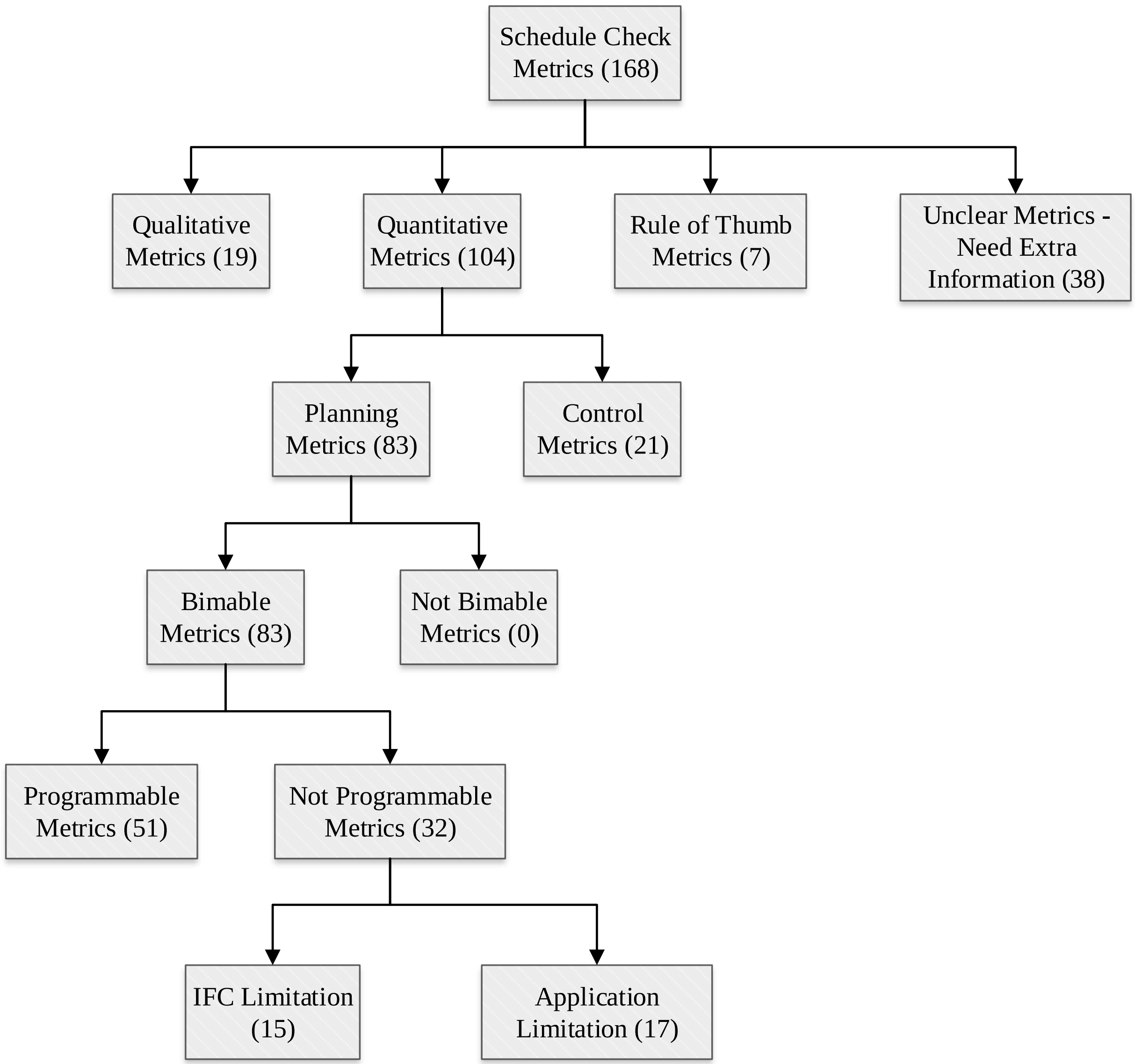

Figure 2 Classification of schedule check metrics

3.2 4D IFC

There are many 3D and 4D modeling software applications available in the industry. Almost all the 3D modeling software applications support IFC export and all the 4D modeling software applications facilitate IFC import. To receive 4D IFC, the 4D modeling software applications should facilitate IFC export. The technological review was conducted to find 4D software which exports 4D IFC. There is only Synchro 4D modeling software available now which supports IFC export for the 4D model. Thus, we utilized Revit for 3D modeling and Synchro for the 4D modeling in this research. The 3D model of sample office building is develop using Revit and exported to IFC2x3 Coordination View 2.0 format. Figure 3 depicts the sample office building. This 3D IFC file was then imported to Synchro and arbitrary schedule was linked to generate the 4D model. Then, 4D model was exported to IFC format to receive 4D IFC.

Figure 3 Sample office building

3.3 Algorithms

After receiving 4D IFC, algorithms were prepared for unique metrics of programmable metrics defined above. Unique metrics meaning the one which have unique logic. The metrics which were similar to each other with different data value were ignored. (E.g. one metric has 5% allowable value while other metric has 10% allowable value). So, finally 22 metrics were selected and their algorithms were made.

3.3.1 Procedure for making algorithms

To get general idea and understanding of IFC files, a third-party application called “STEP File Analyzer” was used. This software exports CSV and Excel files which helps you interpret data more clearly.

Algorithms were written in Pseudocode. “Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a programming language but is intended for human reading rather than machine reading.” [10]. Because Pseudocode has no specific Format, we defined our format. The Table 5 explains meaning of certain symbols and words.

Table 5 Pseudocode terms

Pseudocode terms	**Description (Meaning)**
X == Y	If X equals Y
X != Y	If X Not equals Y
X > Y	If X greater than Y
BEGIN FUNCTION	It begins an independent function which will be used in some other functions when necessary
BEGIN MAIN	It is the main (core) function of that particular metric
" "	Anything between " " these brackets will be printed exactly same.
PRINT "Message"	It will print word Message
PRINT Name	It will print whatever information is stored in Name

[ ]	Anything between [ ] these brackets is giving you information
Task [5]	Go to **5th** row in List called Task
Task [i]	Go to **i** row, where i is generally a variable like x
X [ID]	The type of data in **X** are ID
X [Name]	The type of data in **X** are Name
DECLARE	Create new
DECLARE LIST: X	Create new LIST called **X**
DECLARE NUMBER: Y	Create new Number called **Y**
STRING	A **STRING** is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks)
DOUBLE	Number with decimal points
IfcTask.Name or X.Object	Anything after dot (.) is an attribute or subclass or subtype. e.g. IfcTask is a class and Name is an attribute, X is a list and Object is a subtype
IfcTask.Name[i]	Go to **IfcTask** and Pick the **Name** in **i** row
If X.Name[i] == Y.Name[j]	When **i** row of **Name** in **X** equals **j** row of **Name** in **Y**
INPUT	What are the specific inputs required in function (along with attribute)
OUTPUT	What will be the outputs of function
OUTPUT and PRINT are not Same.	

3.3.2 Explanation of Pseudocode Lines

Certain lines of Pseudocodes are explained below to increase readers understanding.

BEGIN FUNCTION TotalTasks (IfcTask, IfcRelNests)

The Name of function is TotalTasks. IfcTask and IfcRelNests are inputs required for Function TotalTasks.

BEGIN MAIN TasksWithMissingLogic (IfcRelSequence, IfcTask, AllowablePercent = 5)

The Name of function is TasksWithMissingLogic. Third input is User input where it has a default value of 5 if there is no input from User.

BEGIN FUNCTION Example (IfcTask, A = 50, B = 30)

Name of function is Example. A and B are user inputs.

FOR EACH ID IN IfcTask

Here you run a loop and use Each ID in IfcTask one by one.

ADD IfcRelNests.RelatingObject[ID] to List B

ADD data of IfcRelNests.RelatingObject in List B. [ID] represents that the data stored in IfcRelNests.RelatingObject are IDs.

NumberOfMissingLogic = COUNT ID in MissingLogic

Here, you count all the IDs in MissingLogic and Store them in “NumberOfMissingLogic”.

X = CALL FUNCTION TotalTasks (IfcTask, IfcRelNests)

You are calling function called “TotalTasks”. IfcTask and IfcRelNests are the default inputs required for running this function. You will store the output of function in List called “X”.

BEGIN FUNCTION MeasureCheck

''' INPUT NUMBER: Numerator, Denominator

INPUT DOUBLE: value

INPUT STRING: Condition, MetricName

OUTPUT STRING: Message '''

This function requires five input called Numerator, Denominator, value, Condition, MetricName. This function will give output called Message.

DECLARE NUMBER: Z = Numerator * 100/Denominator

You declare a number called Z which has a formula.

PRINT CALL FUNCTION MeasureCheck (Numerator: 5, Denominator: 100, value: 5, Condition: >, MetricName: "XYZ")

You are calling a function with these five inputs and printing the output.

IF (Z Condition value)

RETURN “NOT Satisfied”

ELSEIF

RETURN “Satisfied”

END IF

Now, If (Z: 10, Condition: >, value: 5). Function will run like If (10 > 5) and will give output “Not Satisfied”.

PRINT "Measure Not Satisfied." + MetricName + Z

If MetricName = Logic & Z = 10. Output will be “Measure Not Satisfied. Logic = 10”.

DECLARE LIST TEMP [Name, TaskID]

Creating a new list called “TEMP” which has two sub-types called “Name” and “TaskID”.

3.3.3 Algorithm Samples

Algorithms (Appendix B) are written to run as a single program. In this program, there are two types of functions i.e. Global and Main. Global functions refer to individual (independent) functions which are meant to be called by other Main functions when needed. While each Main function refers to one particular metric, meaning it is the core function of that metric (E.g. 22 metrics refer to 22 Main functions). Figure 4 and 5 are examples of global functions and example of the main function is given in Figure 6.

```
1. Total Tasks:

BEGIN FUNCTION TotalTasks(IfcTask, IfcRelNests)
    '''  INPUT: IfcTask.ID(),
                IfcRelNests.RelatingObject()
         OUTPUT: TotalTasks - A list contaning all Tasks'''

    DECLARE LIST: A, B, TotalTasks

    FOR EACH ID IN IfcTask
        ADD IfcTask.ID to List A
    END FOR

    FOR EACH IfcRelNests.RelatingObject[ID]
        ADD IfcRelNests.RelatingObject[ID] to List B
    END FOR

    TotalTasks = List A - List B

    RETURN TotalTasks
END
```

This global task will give output as IfcTask IDs of all task whenever being called

Figure 4 Global function for Total Tasks

```
3. Measure Check:

BEGIN FUNCTION MeasureCheck
    '''  INPUT: NUMBER Numerator, Denominator
         INPUT: DOUBLE value
         INPUT: STRING Condition, MetricName
         OUTPUT: Message '''

    DECLARE STRING: Message
    DECLARE NUMBER: Z = Numerator * 100/Denominator;

    IF(Z Condition value)
        RETURN Message = [Red color]"Measure Not Satisfied." + MetricName + Z;
    ELSE
        RETURN Message = [Green color]"Measure Satisfied." + MetricName + Z;
    END IF
END
```

All the required inputs

Evaluating the condition

Message that will be returned

Figure 5 Global function for Measure Check

8. Logic Density

Relationships per activity or Relationship ratio or Logic Density = # of Logic links / # Total Tasks

```
BEGIN MAIN LogicDensity (IfcRelSequence, AllowableValue = 2.1 )

    '''  INPUT: IfcRelSequence.ID()

    DECLARE LIST: TotalTasks
    DECLARE NUMBER: TotalNumberOfTask = 0, LogicLinks = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks
    TotalNumberOfTask = 100 * TotalNumberOfTask

    LogicLinks = COUNT ID in IfcRelSequence

    PRINT CALL FUNCTION MeasureCheck(Numerator: LogicLinks, Denominator: TotalNumberOfTask, value:
    AllowableValue, Condition: >, MetricName: "LogicDensity%=")
END
```

This is user input

Calling a Global function called “TotalTask”

Calling and Printing the output of Global function called “MeasureCheck”

Figure 6 Example of Main function

4 MODEL VALIDATION

As there is no working model, you cannot validate the model. So, As discussed earlier, we used a third-party application called “STEP File Analyzer” to generate CSV and Excel files which helps you interpret data more clearly. Using these excel files, errors were validated in IFC files. We created various scenarios to validate errors. A scenario meaning, a case where an error is induced in the model intentionally to validate the data. An example is discussed below to create better understanding.

Example: a task called Foundation has a duration of 5 days (144000 seconds) considering normal work day of 8 working hours (Figure 7). The duration of this task was increased to 60 days (1728000 seconds) to validate the error in IFC file (Figure 8).

1	IfcScheduleTimeControl (41)					
2						
3	ID	GlobalId	OwnerHistory	ScheduleDuration	Name	Description
4	48701	3NokVVgsH6aRUs2p$5W5_4	IfcOwnerHistory 5	1814400		
5	48710	3gC0$xPQT7495FmP7p_$Dq	IfcOwnerHistory 5	230400		
6	48719	3MDQqkX_969PX5xnlX79sB	IfcOwnerHistory 5	144000		
7	48728	0mHop__Yj1OQGrytDRxwnR	IfcOwnerHistory 5	0.00E+00		

Figure 7 Foundation task with 5 days duration

1	IfcScheduleTimeControl (41)					
2						
3	ID	GlobalId	OwnerHistory	ScheduleDuration	Name	Description
4	48701	3NokVVgsH6aRUs2p$5W5_4	IfcOwnerHistory 5	1814400		
5	48710	3gC0$xPQT7495FmP7p_$Dq	IfcOwnerHistory 5	230400		
6	48719	3MDQqkX_969PX5xnlX79sB	IfcOwnerHistory 5	1728000		
7	48728	0mHop__Yj1OQGrytDRxwnR	IfcOwnerHistory 5	0.00E+00		

Figure 8 Foundation task with 60 days duration

How Algorithm will work: Algorithms is broken in parts to explain process clearly.

```
7. Number of Activities with High duration
High Duration % = total # of tasks with high Duration × 100 / # of Total tasks

BEGIN MAIN TasksWithHighDuration (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask,
HighDurationValue = 44, AllowablePercent = 5)

    '''   INPUT: IfcRelAssignsTasks.RelatedObjects()
                 IfcRelAssignsTasks.TimeForTask()
                 IfcScheduleTimeControl.ID()
                 IfcScheduleTimeControl.TotalFloat()
                 IfcTask.ID()
                 IfcTask.Name()
                 IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, Temp1, Temp2, HighDuration
    DECLARE NUMBER: TotalNumberOfTask = 0, NumberOfHighDuration = 0
```

Figure 9 Algorithm part 1

Following are the steps

- The Function called TasksWithHighDuration is created.
- IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask are classes required as input
- Two user defined number called HughDurationValue and AllowablePercent with their default values are also defined as input in this function.
- Then, specific inputs are defined. (i.e. IfcRelAssignsTasks.RelatedObjects() where RelatedObjects is an attribute of class IfcRelAssignsTasks)
- Four new list are created
- Two New Numbers are created.

```
TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
TotalNumberOfTask = COUNT ID in TotalTasks

HighDurationValue = HighDurationValue * WorkingHours * 3600

FOR EACH ID IN IfcScheduleTimeControl
    IF(IfcScheduleTimeControl.ScheduleDuration[i] > HighDurationValue)
        ADD IfcScheduleTimeControl.ID[i] IN Temp1
    END IF
END FOR
```

Figure 10 Algorithm part 2

1	IfcScheduleTimeControl (41)					
2						
3	ID	GlobalId	OwnerHistory	ScheduleDuration	Name	Description
4	48701	3NokVVgsH6aRUs2p$5W5_4	IfcOwnerHistory 5	1814400		
5	48710	3gC0$xPQT7495FmP7p_$Dq	IfcOwnerHistory 5	230400		
6	48719	3MDQqkX_969PX5xnlX79sB	IfcOwnerHistory 5	1728000		
7	48728	0mHop__Yj1OQGrytDRxwnR	IfcOwnerHistory 5	0.00E+00		

Figure 11 IfcScheduleTimeControl class

- A Global Function called TotalTask is called and stored in list called “TotalTasks”.
- Total number of tasks is counted and stored in number called “TotalNumberOfTask”
- HighDurationValue is converted into seconds
- Then For loop is run to identify tasks whose Schedule Duration is more than HighDurationValue (default = 44 days)
- If the condition is satisfied, their IfcScheduleTimeControl IDs are stored in “Temp1” list.

```
FOR EACH ID IN Temp1
    FOR EACH IfcRelAssignsTasks.TimeForTask
        IF (IfcRelAssignsTasks.TimeForTask[i] == Temp1)
            ADD IfcRelAssignsTasks.RelatedObjects[i] to Temp2
        END IF
    END FOR
END FOR
```

Figure 12 Algorithm part 3

IfcRelAssignsTasks (41)						
ID	Name	Descri	RelatedObjects	RelatedObjectsType	RelatingControl	TimeForTask
48708			(1) IfcTask 48700		IfcWorkSchedule 48686	IfcScheduleTimeControl 48701
48717			(1) IfcTask 48709		IfcWorkSchedule 48686	IfcScheduleTimeControl 48710
48726			(1) IfcTask 48718		IfcWorkSchedule 48686	IfcScheduleTimeControl 48719
48735			(1) IfcTask 48727		IfcWorkSchedule 48686	IfcScheduleTimeControl 48728

Figure 13 IfcRelAssignsTasks class

- Now using other FOR loop, data from RelatedObjects of IfcRelAssignsTasks is stored in "Temp2" list.
- RelatedObjects has Task IDs as shown in Figure 13.

```
FOR EACH ID IN Temp2
    FOR EACH ID IN TotalTasks
    IF (TotalTasks[ID] == Temp2[ID])
        ADD one to NumberOfHighDuration
        ADD ID IN HighDuration
    END IF
END FOR

PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfHighDuration, Denominator:
TotalNumberOfTask, value: AllowablePercent, Condition: >, MetricName: "TasksWithHighDuration%=")
```

Figure 14 Algorithm part 4

- Now, lastly these tasks are verified whether they are real activities or not.
- If yes, their ids are stored in list called "HighDuration".

- After that, MeasureCheck function is called and its output is printed.

```
    FOR EACH ID IN HighDuration
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] == HighDuration[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

Figure 15 Algorithm part 5

IfcTask (41)			
ID	**Name**	**TaskId**	**Description**
48700	Office building	1	1
48709	Sub-structure	1.1	1.1
48718	Foundation	1.1.1	1.1.1

Figure 16 IfcTask class

- Now, Name and TaskID are printed of tasks with high duration.

5 LIMITATION

5.1 Software application limitation

There are many 4D modeling software applications available in the industry but only Synchro pro software supports the 4D IFC export. However, the current version of Synchro pro software is only compatible with the IFC2x2 and IFC2x3 file format [11]. So, when the 3D model exported in latest version of IFC (i.e., IFC4) is used to import in Synchro, there are lot of errors in data representation of 3D model like missing walls, orientation change of model elements. So, IFC2x3 file format is used to import in Synchro. Furthermore, when 4D IFC is exported from the Synchro, some of the IFC classes (e.g. IfcResource, IfcCostValue, IfcConstraint) related to constraints, resources and cost data are missing, so it is not possible to formulate the model for the measures that are used to check the schedule against constraints, resource and cost. Even though Synchro exports the 4D IFC, there are errors in representation of schedule data in IFC file. For example, as given in the Figure 7, the value of FreeFloat and TotalFloat is zero for all the activities in the schedule. In other words, even the Synchro calculates the free float and total float, IFC file shows wrong data. The value of the IsCritical is False for all the activities even if the critical activities present in the schedule. Thus, 4D IFC exported from Synchro is erroneous, not completely accurate.

	A	R	S	T
1	IfcScheduleTimeControl (41)			
2				
3	ID	FreeFloat	TotalFloat	IsCritical
4	48701	0.00E+00	0.00E+00	FALSE
5	48710	0.00E+00	0.00E+00	FALSE
6	48719	0.00E+00	0.00E+00	FALSE
7	48728	0.00E+00	0.00E+00	FALSE
8	48737	0.00E+00	0.00E+00	FALSE
9	48753	0.00E+00	0.00E+00	FALSE
10	48762	0.00E+00	0.00E+00	FALSE
11	48779	0.00E+00	0.00E+00	FALSE
12	48788	0.00E+00	0.00E+00	FALSE
13	48797	0.00E+00	0.00E+00	FALSE
14	48812	0.00E+00	0.00E+00	FALSE
15	48827	0.00E+00	0.00E+00	FALSE
16	48842	0.00E+00	0.00E+00	FALSE
17	48857	0.00E+00	0.00E+00	FALSE
18	48870	0.00E+00	0.00E+00	FALSE

Figure 17 Error in 4D IFC exported from Synchro

5.2 IFC limitation

There are lot of developments going on IFC versions, the latest version currently available is IFC4. The major limitation of IFC is types of task described in the IFC schema. The types of task available in IFC are attendance, construction, demolition, dismantle, disposal, installation, logistic, maintenance, move, operation, removal, renovation [12]. The information about Level of Effort (LOE) task, resource dependent task, time dependent task is not available in IFC schema. So, it is not possible to distinguish two task types, e.g. it is not possible to identify LOE tasks from the list of large number of activities in the schedule.

6 CONCLUSION AND RECOMMENDATION

6.1 Conclusion

The conceptual framework of criteria, measures and metrics has been developed from the previous literature and industry software tool which are used to assess the quality of the schedule. All the schedule check metrics have been classified into quantitative, qualitative and rule of thumb to identify the number of quantifiable metrics that can be fully automated in the assessment model. The 4D model has been developed to receive 4D IFC model. Third party application STEP File Analyzer was used to generate the CSV (comma separated values) file of IFC in order to understand the structure of the IFC schema. The schedule quality assessment model has been developed in pseudocode that evaluate the schedule in 4D IFC environment against identified quantitative metrics in conceptual framework. After that model was validated by testing schedule with known errors. The developed model uses 4D IFC file to review schedule that makes possible to evaluate the schedule in 4D environment and so not have to depend on 2D time schedule which is difficult and time consuming to understand.

6.2 Recommendation

The current schedule quality assessment model could be written in programming language to develop computer application. The current research focuses only on schedule so, checks related to geometry, e.g., workspace requirement for labours and overall construction process can be added to improve efficiency on-site. The research further can be expanded to check time space conflict using IFC schema to improve the productivity of the overall construction process. The checks related to construction schedule activities assignment to model elements can be combined to review linking of schedule with 3D model. Due to lot of development in IFC schema is going on, issues related to logistic planning, in-site movement, access to the different locations within site may be addressed in future.

REFERENCES

[1] Royal Architectural Institute of Canada, "BIM Explained," 2016. Available: https://www.raic.org/raic/bim-explained. [Accessed 31 October 2018].

[2] J. Steel, R. Drogemuller and B. Toth, "Model Interoperability in Building Information Modeling," *Software & Systems Modeling,* vol. 11, no. 1, pp. 99-109, 2012.

[3] S. F. Moosavi, "Assessment and Evaluation of Detailed Schedules in Building Construction," MASc Thesis, Concordia University, Montreal, 2012.

[4] M. A. Bragadin and K. Kahkonen, "Schedule health assessment of construction projects," *Construction Management and Economics,* vol. 34, no. 12, pp. 875-897, 2016.

[5] S. F. Moosavi and O. Moselhi, "Review of Detailed Schedules in Building Construction," *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction,* vol. 6, no. 3, pp. 05014001- 1 to 9, 2014.

[6] V. Bansal and M. Pal, "Construction schedule review in GIS with a navigable 3D animation of project activities," *International Journal of Project Management,* vol. 27, p. 532–542, 2009.

[7] BuildingSMART, "Open Standards - the basics," 2018. Available: https://www.buildingsmart.org/standards/technical-vision/open-standards/. [Accessed 30 October 2018].

[8] M. Laakso and A. Kiviniemi, "The IFC standard - a review of history, development, and standardization," *Journal of Information Technology in Construction,* vol. 17, pp. 134-161, 2012.

[9] V. K. Bansal, "Use of GIS and Topology in the Identification and Resolution of Space Conflicts," *Journal of Computing in Civil Engineering,* vol. 25, no. 2, pp. 159-171, 2011.

[10] Viking Code School, "Viking Code School Prep," 2018. Available: https://www.vikingcodeschool.com/software-engineering-basics/what-is-pseudo-coding. [Accessed 22 September 2018].

[11] Synchro Software, "Integrations," 2018. Available: https://www.synchroltd.com/integrations/. [Accessed 10 October 2018].

[12] BuildingSMART, "IfcTaskTypeEnum," 2018. Available: http://www.buildingsmart-tech.org/ifc/IFC4/Add2/html/schema/ifcprocessextension/lexical/ifctasktypeenum.htm. [Accessed 15 October 2018].

[13] U. D. C. M. A. DCMA, "Earned Value Management System (EVMS) Program Analysis Pamphlet (PAP)," Department of Defence, 2012.

[14] Steelray Project Analyzer, "Steelray Project Analyzer Criteria Library Reference," 2018. Available: https://steelray.com/support_center_analyzer_manual/criteria_library.php. [Accessed 4 June 2018].

[15] Schedule Inspector, "Schedule Inspector User Guide," 2018. Available: https://www.barbecana.com/wp-content/uploads/2016/02/ScheduleInspectorUserGuide.pdf. [Accessed 4 June 2018].

[16] Schedule Analyzer, "Ron Winter Consulting LLC," 2018. Available: http://www.scheduleanalyzer.com/sa_brochure.htm. [Accessed 5 June 2018].

[17] Deltek Acumen Fuse, "What Your Schedule Isn't Telling You," 2018. Available: http://images.more.deltek.com/Web/DeltekInc/%7Ba61e4d20-2eac-4062-b457-afbec4ec6739%7D_White-Paper-What-your-Schedule-Isnt-Telling-You-PPM.pdf?elqTrackId=b2c83df129da4635af9ba0ddb3fa5988&elqaid=4851&elqat=2. [Accessed 2 June 2018].

[18] Deltek Open Plan, "Powerful Enterprise Planning and Scheduling," 2018. Available: https://www.deltek.com/~/media/product%20sheets/open%20plan/open-plan-brief.ashx?la=en. [Accessed 2 June 2018].

[19] Schedule Cracker, "Is your Schedule Ready for the 14-Point DCMA?," 2014. Available: http://schedulecracker.com/wp-content/uploads/2014/07/Schedule_Cracker_Is_Your_Schedule_Ready_for_DCMA.pdf. [Accessed 2 June 2018].

[20] P6 Schedule Checker, "Running the “Check Schedule” Feature with P6 EPPM," 2018. Available: http://www.projwebsite.com/publications/TechTip_Running_the_Check_Schedule_Feature_with_P6_EPPM.pdf. [Accessed 3 June 2018].

[21] XER Reader, "XER Reader is a free tool to run DCMA 14 point checks," 2018. Available: https://www.planacademy.com/xer-reader-free-tool-run-dcma-14-point-checks/. [Accessed 3 June 2018].

[22] XER Schedule Toolkit, "User Manual Overview & Function," 2018. Available: https://xertoolkit.com/support/xer8-user-manual.pdf. [Accessed 3 June 2018].

[23] M. A. Bragadin, "Quality Driven Scheduling in Construction," Tampere University of Technology, 2018.

[24] Deltek Acumen Fuse, "Schedule Quality: 9 Industry Benchmarks for Sound Scheduling," 2018. Available: https://www.deltek.com/~/media/infographics/schedule%20quality.ashx?la=en. [Accessed 2 June 2018].

[25] M. Krzeminski, "Chosen criteria of construction schedule evaluation," *Procedia Engineering*, vol. 153, pp. 345-348, 2016.

[26] B. Garcia de Soto, A. Rosarius, J. Rieger, Q. Chen and B. T. Adey, "Using a Tabu-search Algorithm and 4D Models to Improve Construction Project Schedules," *Procedia Engineering*, vol. 196, pp. 698-705, 2017.

[27] M. A. Bragadin and K. Kahkonen, "Safety, space and structure quality requirements in construction scheduling," *Procedia Economics and Finance*, vol. 21, pp. 407-414, 2015.

[28] Bansal, V.K.; Pal, Mahesh, "Generating, Evaluating, and Visualizing Construction Schedule with Geographic Information Systems," *Journal of Computing in Civil Engineering*, vol. 22, no. 4, pp. 233-242, 2008.

[29] M. Krzeminski, "Construction Team Downtime Minimization Model Including Efficiency Coefficients," *Procedia Engineering*, vol. 161, pp. 18-22, 2016.

[30] M. Kassem, N. Dawood and R. Chavda, "Construction workspace management within an Industry Foundation Class-Compliant 4D tool," *Automation in Construction*, vol. 52, pp. 42-58, 2015.

[31] A. Marx and M. Konig, "Modeling and Simulating Spatial Requirements of Construction Activities," Bochum, Germany, 2013.

APPENDIX A: CONCEPTUAL FRAMEWORK

Conceptual framework of criteria, measures and metrics from previous literatures and industry schedule review tools.

Criteria	Measures	Metrics	References
Network and logic	Number of activities with Missing logic (No predecessors or successors)	Any incomplete task that is missing a predecessor and/or a successor is included. The number of tasks without predecessors and/or successors should not exceed 5%. Missing Logic % = # of tasks missing logic*100/ # of incomplete tasks.	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22]
		Relative weight = 21: No open-ended activity is allowed (activity without predecessor or successor) Compares activity relationships of the schedule with typical activity relationships used in reinforced concrete framing building incorporated in SAE software application and finds missing logic	[3] [23]
		Total Missing Successor Tasks / Total non-Summary Tasks	[14]
		Total Missing Predecessor Tasks / Total non-Summary Tasks	[14]
		19%, Average percentage of activities with missing logic	[24]
		Number of Activities with zero predecessors or zero successors	[16] [22]
	Number of activities with high duration	Number of tasks with high duration should not exceed 5%. Incomplete tasks with a baseline duration greater than 44 working days (2 months), and has a baseline start date within the detail planning period or rolling wave are counted. $\text{High Duration \%} = \frac{\text{Total \# of incomplete tasks with high duration}}{\text{Total \# of incomplete tasks}} \times 100$	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
		It will count the activities which has high duration (> 20 days)	[16]
		12%, Average percentage of activities that have a duration greater than 10% of the project duration	[24]

	Relationship Type (FS, SF, SS, FF)	Finish-to-Start (FS) relationship type should account for at least 90% of the all relationship types being used $$\% \text{ of FS Relationship Types} = \frac{\# \text{ of logic links with FS Relationships}}{\# \text{ of logic links}} \times 100$$	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22]
		Start to Finish (SF) relationship is counter-intuitive, it should be avoided	[23] [16]
	Relationships per activity/ Relationship ratio/ Logic Density	Relative weight = 27: Total number of relationships/total number of activities should be limited Maximum 2.1 (1.6 per activity, O'brien and Plotnick)	[3] [23]
		3.02%, Average number of Relationships per activity	[24]
	Summary tasks with logic (successors and predecessors)	It should be zero; Summary tasks with logic relationships should be avoided	[23]
		Summary tasks should not have predecessors or successors. Total Summary with Successor Tasks / Total Summary Tasks	[15] [14]
	Empirical rules for starting of activities	Duration of foundation activities is approximately 5% of framing activities duration -Typically once more than 30% of foundation is performed, framing activities can start -Duration of framing activities is approximately 35% of project duration -Typically once framing of three floors is performed, curtain wall activities could start -Duration of curtain wall activities is approximately 30% of project duration -Typically once 30% of curtain wall is performed, architectural activities start -Duration of architectural activities is approximately 40% of project duration -Typically HVAC and electrical activities could start at the same time, once 30% of framing is performed -Duration of electrical activities is approximately 60% of project duration -Duration of HVAC activities is approximately 65% of project duration -Duration of firefighting activities is approximately 30% of project duration -Typically once framing is done, elevator & escalator activities start -Duration of elevator & escalator activities is approximately 30% of project duration	[5]
	Number of Dangling Activities	Discovers tasks that are missing links to its start or finish points. Dangling Tasks / Incomplete Effort Tasks	[16] [14]

	Tasks Without Finish-to-Start Predecessors	Total Non-FS Predecessor Tasks / Incomplete Effort Tasks	[14]
	Number of activities with high predecessors	This test counts the number of tasks that have more than a specified number of predecessors. This number can be set by the user, the default being 10 predecessors.	[15]
		10%, Average percentage of activities that have a high number of predecessors	[24]
	Number of activities with high successors	This test counts the number of tasks that have more than a specified number of successors. This number can be set by the user, the default being 10 successors.	[15]
	Tasks with redundant predecessors	Schedule Inspector counts the number of tasks that have redundant predecessors. This test relates to relationships that have no effect because they are implied by other logic.	[15]
	Number of Activity without finish relationship	Counts the number of activities	[16]
	Number of bogus activity Relationship Records	Counts the number of activities	[16]
Resources	Number of activities with missing resource	Tasks with incomplete resource information are counted. Count should be zero. Missing resources% = (total # of incomplete tasks with missing resource)/ (total # of incomplete task) x 100 A quality schedule should have resources assigned to every task (all task with durations greater than zero have resources, people or costs, assigned).	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
		Relative weight = 24: Schedule should be loaded with resources Relative weight = 25: A responsible party/person should be assigned to each activity	[3]

	Trades' Rate of completion per week	Relative weight = 21: Compliance of each trade’s progress curve (Rate of completion per week) with historical (typical) average Data according to total installed cost and phase duration	[3] [23]
	Resource productivity	production/time: RS Means building construction cost data, > 30% deviation not allowed	[3]
		The resources required to complete each activity, including their availability and productivity, should be considered to determine the duration of each activity, whether the resources are applied to activities in the schedule model. - The schedule should contain the most current resource estimates available	[23]
	Trades' Peak Resource Loading	Relative weight = 23: Compliance of peak resource loading of each trade with historical average data according to total installed cost and phase duration	[3] [23]
	Peak to average labor ratio	Relative weight = 20: Peak to average number of laborers for each trade should comply with the average historical data according to total installed cost and phase duration	[3] [23]
	Crew size (Number of laborers)	RS Means building construction cost data, > 30% deviation not allowed	[3]
	Brigade downtime (Summed Slack time (ST))	$B^{(n)} = ST^{BS} - ST^{nS}$ $B^{(n)\%} = \left(\frac{ST^{BS} - ST^{nS}}{ST^{BS}}\right) * 100\%$	[25]
	Resource fluctuation	Resource Moment (M) = Σ(r^2/2)	[26]
	Summary tasks With Resources	Summary tasks should not have resources assigned to them. Total Summary with Resource Tasks / Total Effort Tasks	[15] [14]
	Number of resource dependent activities without resources	Counts resource dependent activities without resources, these activities should be set to time dependent and be given a fixed time duration.	[16]

	Number of Time dependent activities with resources	Counts the number of activities	[16]
	Resource lags	Resource lags represent the delay between when the activity starts, and the resource starts. Resources should not start after the task ends.	[16]
	Schedule Leveling	Relative weight = 25, Schedule should be leveled	[3] [23]
	Trades' Peak Resource Loading relation	Relative weight = 22, The relationship between various trades' peak resource loading should follow the historical average trend according to total installed cost and phase duration	[3] [23]
Critical path	Critical path length index (CPLI)	Project schedule's Critical Path Length (CPL) is the length in work days from time now until the next project milestones that is being measured. A CPLI less than 0.95 is a flag. Critical Path Length Index (CPLI) = (CPL + TF) / CPL The CPLI value should be greater than 1.0	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
	Critical path test (Checks continuity of critical path)	Critical path should be continuous through the network, i.e. there must not be broken logic which is the result of missing predecessor and / or successor on task where they are needed. The schedule passes the test if the project finish date matches the added delay into the remaining duration. Critical path test result should be 0 %.	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
	Schedule Criticality rate	Nos (or duration) of critical activities/Nos (or duration) of activities Maximum 25% (25%, O'brien and Plotnick) Relative weight = 28 (when schedule criticality rate base on # critical activities) Relative weight = 24 (when schedule criticality rate base on duration of critical activities)	[3] [23]
		Maximum 30%	[16]
		23%, Average percentage of critical activities	[24]
	Nos of critical activities with out-of-range duration	Maximum 30 days (Critical activities, to be well manageable, should have a duration limited to one pay period, De la Garza)	[3]
		Critical activities should be more detailed (1 to 20 days)	[23]

	Project duration	$A^{(n)} = TT^{BS} - TT^{nS}$ $A^{(n)\%} = \left(\frac{TT^{BS} - TT^{nS}}{TT^{BS}}\right) * 100\%$	[25]
		Project duration = Σ duration of critical activities, D = (a+4m+b)/6, SD = (b-a)/6, Beta Distribution	[26]
	Near criticality rate	Nos of near critical activities/Nos of activities Maximum 10%, Near critical activity has TF < 5 to 10 Relative weight = 23: Number of near critical activities/total number of activities should be limited	[3] [23]
		Maximum 50%, Near critical activity has total float less or equal to 5 days	[16]
	Critical Activity Affiliation	Each critical activity should have a predecessor reflecting physical dependency	[3] [23]
	Multiple critical paths	Managing simultaneous critical path is difficult and should be avoided whenever possible	[23]
	Number of Incomplete Critical Tasks	Total Incomplete Critical Tasks / Total Incomplete Tasks	[14]
	Number of LOE tasks on critical path	Zero, LOE tasks should not be on the critical path.	[15]
Float	Number of activities with negative total float	Ideally, there should not be any negative float in the schedule. An incomplete task with total float less than 0 working days. Negative Float % = (total # of incomplete tasks with negative float)/ (total # of incomplete tasks) x 100	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
		Relative weight = 17: No activity with negative float is allowed	[3]
		8%, Average percentage of activities with negative float	[24]
	Tasks with Total Slack < -20d	Total TS-20 Tasks / Total Incomplete Tasks	[14]
	Tasks with Total Slack < -200d	Total TS-200 Tasks / Total Incomplete Tasks	[14]
	Tasks with Total Slack < 30d	Total TS30 Tasks / Total Incomplete Tasks	[14]

	Tasks with Total Slack > 200d	Total TS200 Tasks / Total Incomplete Tasks	[14]
	Number of activities with out-of-range total float	Percentage of tasks with excessive total float should not exceed 5%. An incomplete task with total float greater than 44 working days (2 months) are counted. High Float % = (total # of incomplete tasks with high float)/ (total # of incomplete tasks) x 100	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
		Activities with total float bigger than 100 days (44 days GAO 2009, Berg et al. 2009) should be avoided. Relative weight = 20: Activities with excessive Total Float should be avoided	[3]
	Float computation	The amount of time a task can slip before affecting the critical path for activities must be calculated	[23]
Lag & Lead (negative lag)	Number of logic links with leads	Number of logic links with a lead (negative lag) in predecessor relationships for incomplete tasks should be counted. It should be zero. Leads % = # of logic links with lead x 100/ Total # of logic links	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
		3%, Average percentage of activities that are being driven by a lead or negative lag	[24]
	Number of logic links with lags	14%, Average percentage of activities that are being driven by a lag	[24]
		The number of relationships with lags should not exceed 5%. Lags % = # of logic links with lags x 100/ Total # of logic links Lag duration should not be greater than predecessor or successor activity duration	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
	Number of Relationship with odd Lag	Relative weight = 28: Should not be greater than the duration of Predecessor or Successor activity	[3]
		lag duration = Maximum 5 days	[16]
	Number of non-overlapping lags	Counts the number of non-overlapping lags	[16]
Soft & hard Constraints	Number of activities with hard constraints	Maximum number of constrains on activities start or finish should be limited to 5% of total activities Relative weight = 22: Number of constraints on activities start and finish should be limited	[3]
		Total Constrained Tasks / Total Non-Summary Tasks	[14]

		2%, Average percentage of activities that contain a “Must Finish On” or “Must Start On” constraint	[24]
		The number of tasks with hard constraints should not exceed 5%. Hard constraints: Must-Finish-On (MFO), Must-Start-On (MSO), Start-No-Later-Than (SNLT), & Finish-No-Later-Than (FNLT) are counted. Hard constraint % = (total # of incomplete tasks with hard constraints)/ (total # of incomplete tasks) x 100	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
	Soft constraints	Number of Soft constraints: ASAP, ALAP, start no earlier than (SNET), finish no earlier than (FNET). The schedule model calculates early start and late start and finish dates for each activity	[23]
Invalid dates and missed tasks	Number of activities with missing baseline finish date	The number of missed tasks should not exceed 5%. Number of tasks with missing baseline start or finish dates are counted. - Missed % = (total # of tasks with actual/forecast date past baseline date)/ (# tasks with baseline finish dates on or before status date) x 100	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
	Invalid dates (invalid forecast / actual dates)	Invalid dates are activity start and finish dates opposing to project status date (i.e. in the future or in the past). There should not be any invalid dates in the schedule.	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
	Number of activities with missing baseline finish date	Total Missing Baseline Tasks / Total Non-Summary Tasks	[14]
	Invalid dates (invalid forecast / actual dates)	Actual Start/Finish Dates in the Future: Actual dates should not exist in the future. Total Future Tasks / Non-Summary Tasks Actual Finish Before Actual Start: Total Actual Finish Before Actual Start Tasks / Total Effort Tasks	[14]
	Out of Sequence Tasks	Used to identify tasks that have begun when their predecessors or successors are not 100% complete. Out of sequence task are: Finish-to-Finish: Successor Complete and predecessor not complete, Finish-to-Start: Predecessor not complete and successor started, Start-to-Finish: Successor complete and predecessor not started, Start-to-Start: Successor started, and predecessor not started	[14] [15] [23]

	Total number of resource assignments with misaligned dates	Assignment dates% = 100x Total # of resource assignments with misaligned dates/Total # of resource assignments	[22]
Schedule baseline	Baseline execution index (BEI)	BEI compares the cumulative number of tasks completed to the cumulative number of tasks with a baseline finish date on or before the current reporting period. A BEI less than 0.95 is a flag BEI = (total # of task complete)/ (total # of tasks completed before now + total # of tasks missing baseline finish date) BEI value should be greater than 1.00	[13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23]
		Tasks that have finished / Tasks that should have finished. 1.0 is perfectly on track, < 1.0 worse, > 1.0 better. Above 0.95 green, 0.90 to 0.85 yellow, Under 0.85 red.	[14]
	Tasks Without Baseline	This test counts tasks which do not have one or both baseline dates.	[15]
		if an activity has been deleted within the live project and a new activity created that uses the same activity ID Tasks without B/L%= 100 x (# of tasks in live project - # of corresponding B/L tasks)/ Total# of tasks in live project	[22]
	Schedule maintenance	Schedule should be maintained/updated on a regular basis	[23]
	Actual progress	Actual progress must be compared to the baseline plan - For every schedule change, logic, time and cost impacts should be assessed	[23]
	Variance report	Any variation that exceed predetermined user-defined threshold limits shall be reported. - Easily modified and update variation should be tolerated	[23]
	Total number of tasks with late actual or projected start date	Late task starting % = 100 x Total# of tasks with late actual/project start date vs baseline/ Total# of tasks	[22]
Schedule definition/overview	Project duration	Relative weight = 21: Project duration should conform with parametric scheduling results	[3]
		Compliance with the overall contract start and completion dates	[23]

	Number of Milestones Task	Relative weight = 29: At least two milestones, start & end, should be included in each schedule	[3] [23]
		Total Milestone Tasks / Total Tasks	[14]
		Count the number of milestones	[16]
	Verification of Project Performance	Relative weight = 23: Generated S-Curve should be in compliance with typical S-curves	[3]
		Budgeted Cost for Work Performed (BCWP) / Budgeted Cost for Work for Scheduled (BCWS). Greater than 1.0 is favorable. Less than 1.0 is unfavorable.	[14]
	Calendar Verification	Relative weight = 27: Non-working days should be indicated in the project calendar	[3]
	Phase Duration	Relative weight = 21: Each phase duration (Engineering, procurement, ...) should be in compliance with historical average data according to Total Installed Cost	[3]
	Phase Overlap	Relative weight = 22: Engineering should not overlap construction by more than a percentage	[3]
	Working Hours Schedule-Estimate Compliance	Relative weight = 28: Basis of scheduling should be in compliance with basis of estimate as regards working hours	[3]
	Master schedule & Critical path ID	Master schedule and critical path must be identified	[23]
	Realistic network logic	Schedule reflects realistic and meaningful network logic / phasing and sequencing	[23]
	Schedule logic integration	Schedule logic should be vertically and horizontally integrated Vertical schedule integration violation: all tasks should flow up to their summary task	[23]
	Number of Milestones with resources	Milestones do not represent actual work and should not be resource loaded	[15]

Workspace	Congestion Index (labor density)	Relative weight = 22: Maximum number of workers per square meter should be limited to avoid congestion (25 to 30 sqm/man, 200 sq./person)	[3] [23]
	Safe & non-congested work areas	Subjective	[27] [23]
Activity definition	Activity easy to monitor	Activities can be easily monitored and measured (i.e. they are real)	[23]
	Activity name understandable	Activity name is understandable/usable All the activities provide the total scoping of the project/each element of the project	[23] [16]
	Number of Activity Code definitions are blank	Counts the number of activities	[16]
	Number of Activity Code definitions are duplicated	Counts the number of activities	[16]
	Number of Activity Code definitions are insufficiently defined	Counts the number of activities	[16]
	Number of Effort Tasks	Total Effort Tasks / Total Tasks	[14]
	Number of inactive tasks	Identifies inactive tasks. Project 2010 introduced a facility whereby you may mark a task inactive.	[15]
	Number of Incomplete Tasks	Total Incomplete Tasks / Total Effort Tasks	[14]

	Number of manually schedule tasks	Counts the number of activities. Project 2010 allows tasks to be flagged as manually scheduled, which defeats the purpose of CPM scheduling.	[15]
	Number of Summary Tasks	Total Summary Tasks / Total Tasks	[14]
	Number of tasks with duplicate name	Name of task should be unique	[15] [16]
	Out of range activity numbers	Relative weight = 25: If the number of activities has not been indicated in the contract, it should be within a min/max range 50 to 1000 (at least one activity for each 10,000 $, O'brien and Plotnick) (40 to 250, De la Garza)	[3] [23] [16]
	Special activities included	Special activities should be included -Start up and testing activities -Procurement activities -Submittal review activities - Submittal activities -Permits and environmental remediation	[23]
	Submission date	Activity/schedule submission date	[23]
	WBS verification	Relative weight = 28: Scheduling should be based on an approved WBS	[3] [23]
		Number of blank WBS code definitions: All tasks should have a Work Breakdown Structure (WBS) identifier assigned. Total Missing WBS Tasks / Total Tasks	[16] [14]
Activity duration	Continuity of production	Continuity of production in the activity should be maintained	[23]
	Duration < 5d	Total D5d Tasks / Total Incomplete Tasks	[14]
	Duration > 20d	Total D20d Tasks / Total Incomplete Tasks	[14]
	Number of activities with out-of-range duration	Activity duration should be within a min/max range 5 to 90 days (not more than two times the update cycle, ideally never more than 3 times the update cycle, PMI) (44 days, Berg et al.) (5 to 25 days, De La Garza) Relative weight = 20	[3]
		Duration limits range from 1 to 30 days (detailed activity) / planning packages requiring detail planning	[23]

	Remaining duration	In progress activity: Remaining duration < Original duration, Unstarted activity: Remaining duration = Original duration	[16]
Monetary value/cost of activities	Total monetary value	The monetary value/cost of the whole construction schedule should comply with the total contract amount; − The monetary value associated with an activity should play little role in constraining its duration; − The decision to include the cost of materials with the cost of their installation should be based on (a) whether or not the owner wants to reimburse for materials soon after they arrive at site and (b) the ratio of cost of materials to total activity cost;	[23]
	Number of activities with high cost	$\text{High Cost \%} = 100\text{x} \dfrac{\text{Total \# of incomplete tasks with high cost}}{\text{Total \# of incomplete tasks}}$	[22]
	Progress payment	Progress payment requests should be reasonable and based upon scheduled activities which are in progress. - Cash flow front-end loading should be avoided	[23]
	Number of Activity expenses without cost account code	Count the number of activities	[16]
	Number of Activities with negative budgeted expense	Count the numbers, Budgeted Expenses should only be negative if the task is designed to make money.	[16]
	Number of Activities with negative actual costs	Count the numbers, Actual Expenses should only be negative if the task is designed to make money.	[16]
Project total level of effort	Project Effort Ratio	Relative weight = 20: Project critical path effort (number of laborers)/total project effort; should be within a min/max range (0.05 to 0.35)	[3] [23]

	Total amount working hours/days	$C^{(n)} = \sum t_{ij}^{nS} - \sum t_{ij}^{BS}$ $C^{(n)\%} = \left(\frac{\sum t_{ij}^{nS} - \sum t_{ij}^{BS}}{\sum t_{ij}^{nS}}\right) * 100\%$	[25]
		Total amount of working hours/days of project	[23]
Activity progress evaluation	Percentage complete	Assessing progress on an activity should make both the percentage complete and the expected real remaining duration consistent. -Duration-based percent complete should be compared with scope-based percent complete.	[23]
	Schedule slippage	In case of schedule slippage, the entire path to which the lagging activities belong should be monitored (even if contains no-critical activity)	[23]
	Total Number of Incomplete tasks with in-progress error	In progress errors % = 100x $\frac{\text{Total \# of incomplete tasks with in-progress errors}}{\text{Total \# of incomplete tasks}}$	[22]
Activity timing	Weather Sensitive Activities	Relative weight = 26: The duration should reflect the season of the year in which activities are to be executed (if weather sensitive, i.e. its materials and/or labor are affected by either water, temperature or moisture);	[3] [23]
	Building enclosure dependencies	The building enclosure should be logically related to weather-sensitive activities. A building is considered enclosed when weather sensitive-work can proceed, and the building can be heated.	[23]
Complexity of schedule	Level of details for activity	Subjective	[28] [6]
Construction process productivity	Work continuity	Subjective	[23]
	Work-flow of resources	Subjective	[23]
Project cost ratio	Project Cost Ratio range	Relative weight = 21: Project critical path cost/total project cost; should be within a min/max range (0.10 to 0.30)	[3] [23]
Schedule process	Standardized scheduling procedure	The schedule process is adherent to standardized scheduling procedure	[23]
		Relative weight = 30: Schedule should be developed by participation of parties associated with the project	[3]

	Subcontractors Participation	Relative weight = 27: Subcontractors responsible for considerable parts of project should become involved in schedule development having their work integrated and coordinated.	[3] [23]
	Main subcontractor participation	Main subcontractor participation is requested	[23]
	Verification of Subcontractors' Scope of Work	Relative weight = 28: The schedule should reflect the start and completion dates for prime contractors involved	[3]
	Project calendars identification	Project calendars are identified	[23]
	Activity coding structure	Subjective	[23]
Activity mis-assignments	No activity mis-assignments	There should not be any milestones/activity mis-assignment. - Task mis-assignments (a task whose date has finished beyond the milestone to which it is assigned, or is associated with tasks or milestones which are not included in the file) - Orphan tasks (a task that has no association with a milestone, although it might be a successor or predecessor of other tasks)	[23]
	No empty milestones	A milestone that contains less than two tasks, i.e. zero or one	[23]
Activity sequencing	Reasonable activity sequencing	Activities' interdependencies and sequencing should be logical / reasonable. - Activities may be overlapped without affecting production continuity - Tasks with explicit names in logical sequence with right dependencies - Complete specification of logic in order to maintain construction strategy (under all date scenario: early, levelled, late)	[23]
Buffers (Reserves)	Buffers	Buffers should be inserted at the right places	[23]
	Schedule projections	Schedule projections should be based on comparisons between what was planned and what happens	[23]

Schedule projections (conformance)	Corrective actions	Corrective actions or changes must be approved and documented - The interim schedule approval implies the acceptance of a practical and logical way to finish the remaining work on time. - The schedule output should be analyzed for variances and changes to the critical path and completion date	[23]
Schedule scope	Project Scope Definition	Relative weight = 27: All aspects of project scope should be adequately defined before scheduling	[3]
Special considerations	Permits & Environmental Remediation	Relative weight = 27: Permits & environmental remediation should be included in the schedule (if applicable)	[3]
	Procurement Activities	Relative weight = 29: Procurement activities should precede special installation tasks	[3]
	Startup and Testing Activities	Relative weight = 28: Start up and testing activities should be included in the schedule (if applicable)	[3]
	Submittal Activities	Relative weight = 28: Material and/or methods requiring prior approval must have their submittal activities in the network	[3]
	Submittals Review Activities	Relative weight = 27: Submittal reviews to be reflected in schedule as an activity	[3]

APPENDIX B: SCHEDULE QUALITY ASSESSMENT MODEL OR ALGORITHMS

GLOBAL FUNCTIONS

1. Total Tasks:

```
BEGIN FUNCTION TotalTasks(IfcTask, IfcRelNests)
    '''  INPUT: IfcTask.ID(),
                IfcRelNests.RelatingObject()
         OUTPUT: TotalTasks - A list contaning all Tasks'''

    DECLARE LIST: A, B, TotalTasks

    FOR EACH ID IN IfcTask
        ADD IfcTask.ID to List A
    END FOR

    FOR EACH IfcRelNests.RelatingObject[ID]
        ADD IfcRelNests.RelatingObject[ID] to List B
    END FOR

    TotalTasks = List A - List B

    RETURN TotalTasks
END
```

2. Critical Task:

```
BEGIN FUNCTION CriticalTasks (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask)

    '''  INPUT: IfcRelAssignsTasks.RelatedObjects(),
                IfcRelAssignsTasks.TimeForTask()
                IfcScheduleTimeControl.ID(),
                IfcScheduleTimeControl.IsCritical(),
                IfcTask.ID()
         OUTPUT: CriticalTasks '''

    DECLARE LIST: TotalTasks, Temp1, Temp2, CriticalTasks

    FOR EACH ID IN IfcScheduleTimeControl
        IF(IfcScheduleTimeControl.IsCritical == True)
            ADD IfcScheduleTimeControl.ID IN Temp1
        END IF
    END FOR
```

```
    FOR EACH ID IN Temp1
        FOR EACH IfcRelAssignsTasks.TimeForTask
            IF (IfcRelAssignsTasks.TimeForTask == Temp1)
                ADD IfcRelAssignsTasks.RelatedObjects to Temp2
            END IF
        END FOR
    END FOR

    FOR EACH ID IN Temp2
        FOR EACH ID IN TotalTasks
            IF (TotalTasks[ID] == Temp2[ID])
                ADD TotalTasks[ID] to CriticalTasks
            END IF
        END FOR
    END FOR

    RETURN CriticalTasks
END
```

3. Measure Check:

```
BEGIN FUNCTION MeasureCheck
    '''  INPUT: NUMBER Numerator, Denominator
         INPUT: DOUBLE value
         INPUT: STRING Condition, MetricName
         OUTPUT: Message '''

    DECLARE STRING: Message
    DECLARE NUMBER: Z = Numerator * 100/Denominator;

    IF(Z Condition value)
        RETURN Message = [Red color]"Measure Not Satisfied." + MetricName + Z;
    ELSE
        RETURN Message = [Green color]"Measure Satisfied." + MetricName + Z;
    END IF
END
```

MAIN FUNCTIONS - METRICS:

1. Number of activities with Missing logic

Missing logic % = # Tasks with missing logic × 100/ # Total tasks

```
BEGIN MAIN TasksWithMissingLogic(IfcRelSequence, IfcTask, AllowablePercent = 5)
    '''  INPUT: IfcRelSequence.RelatingProcess()
                IfcRelSequence.RelatedProcess()
                IfcTask.ID(),
                IfcRelNests.RelatingObject() '''
```

```
    DECLARE LIST: TotalTasks, MissingLogic, Temp
    DECLARE NUMBER: TotalNumberOfTask, NumberOfMissingLogic

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks

    FOR EACH RelatingProcess[ID] IN IfcRelSequence
        FOR EACH ID IN TotalTasks
            IF (TotalTasks[ID] != RelatingProcess[ID])
                ADD RelatingProcess[ID] in MissingLogic
            END IF
        END FOR
    END FOR

    FOR EACH RelatedProcess[ID] IN IfcRelSequence
        FOR EACH ID IN TotalTasks
            IF (TotalTasks[ID] != RelatedProcess[ID])
                IF (RelatedProcess[ID]!= MissingLogic[ID])
                    Add RelatedProcess[ID] in MissingLogic
                END IF
            END IF
        END FOR
    END FOR

    NumberOfMissingLogic = COUNT ID in MissingLogic

    PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfMissingLogic, Denominator:
    TotalNumberOfTask, value: AllowablePercent, Condition: >, MetricName: "Missing Logic% =")

    FOR EACH item IN Temp
        FOR EACH ID IN IfcTask
            IF (IfcTask.ID == Temp.RelatingProcess)
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF

            IF (IfcTask.ID == Temp.RelatedProcess)
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

Checking whether the measure is satisfied and printing

Printing the tasks that have missing logic

2. Number of Logic links with Lag

Lag % = # of Logic links with lag × 100/ # of logic links

BEGIN MAIN LagLogic (IfcRelSequence, IfcTask, AllowablePercent = 5)

```
    '''  INPUT: IfcRelSequence.TimeLag()
                IfcRelSequence.RelatingProcess()
                IfcRelSequence.RelatedProcess()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: Temp[RelatingProcess, RelatedProcess]
    DECLARE NUMBER: Lag = 0, NoOfLogicLinks = 0

    NoOfLogicLinks = Count Total IfcRelSequence.ID

    FOR EACH TimeLag IN IfcRelSequence
        if(IfcRelSequence.TimeLag[i] > 0)
            ADD one to Lag
            ADD IfcRelSequence.RelatingProcess[i], IfcRelSequence.RelatedProcess[i] to Temp
            END IF
        END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: Lag, Denominator: NoOfLogicLinks, value:
    AllowablePercent, Condition: >, MetricName: "Lag% =")

    FOR EACH item IN Temp
            FOR EACH item IN IfcTask.ID
                IF (IfcTask.ID == Temp.RelatingProcess)
                    PRINT IfcTask.Name, IfcTask.TaskID
                END IF

                IF (IfcTask.ID == Temp.RelatedProcess)
                    PRINT IfcTask.Name, IfcTask.TaskID
                END IF
            END FOR
    END FOR
END
```

Printing the tasks that have missing logic

3. Number of Logic links with Lead

Lead % = # of Logic links with lead × 100/ # of logic links

```
BEGIN MAIN LeadLogic (IfcRelSequence, IfcTask, AllowablePercent = 0)

    '''  INPUT: IfcRelSequence.TimeLag()
                IfcRelSequence.RelatingProcess()
                IfcRelSequence.RelatedProcess()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE NUMBER: NoOfLogicLinks = 0, Lead = 0
```

```
    DECLARE LIST: Temp[RelatingProcess, RelatedProcess]

    NoOfLogicLinks = Count ID in IfcRelSequence

    FOR EACH IfcRelSequence.TimeLag
        if(IfcRelSequence.TimeLag[i] < 0)
            ADD one to Lead
            ADD IfcRelSequence.RelatingProcess[i], IfcRelSequence.RelatedProcess[i] to Temp
        END IF
    END FOR


    PRINT CALL FUNCTION MeasureCheck(Numerator: Lead, Denominator: NoOfLogicLinks, value:
    AllowablePercent, Condition: >, MetricName: "Lead% =")

    FOR EACH item IN Temp
        FOR EACH item IN IfcTask.ID
            IF (IfcTask.ID == Temp.RelatingProcess)
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF

            IF (IfcTask.ID == Temp.RelatedProcess)
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

4. FS Relationship Type

% of FS relationship types = # of logic links with FS relationship × 100/ # of logic links

```
BEGIN MAIN FSRelationship (IfcRelSequence, IfcTask, AllowablePercent = 90)

    '''  INPUT: IfcRelSequence.SequenceType()
                IfcRelSequence.RelatingProcess()
                IfcRelSequence.RelatedProcess()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE NUMBER: NoOfLogicLinks = 0, LinksWithFS = 0
    DECLARE LIST: Temp[RelatingProcess, RelatedProcess]

    NoOfLogicLinks = Count ID in IfcRelSequence

    FOR EACH IfcRelSequence.SequenceType
        if(IfcRelSequence.SequenceType[i] == "FINISH_START")
            ADD one to LinksWithFS
```

```
            ADD IfcRelSequence.RelatingProcess[i], IfcRelSequence.RelatedProcess[i] to Temp
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: LinksWithFS, Denominator: NoOfLogicLinks, value:
    AllowablePercent, Condition: <, MetricName: "FS RelationTypes%=")

    FOR EACH item IN Temp
            FOR EACH item IN IfcTask.ID
                IF (IfcTask.ID == Temp.RelatingProcess)
                    PRINT IfcTask.Name, IfcTask.TaskID
                END IF

                IF (IfcTask.ID == Temp.RelatedProcess)
                    PRINT IfcTask.Name, IfcTask.TaskID
                END IF
            END FOR
    END FOR
END
```

5. Number of activities with out-of-range total float

High Float % = total # of tasks with high float × 100 / # of Total tasks

```
BEGIN MAIN TasksWithOutOfRange (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask, HighFloatValue =
44, AllowablePercent = 5)

    '''   INPUT: IfcRelAssignsTasks.RelatedObjects()
                IfcRelAssignsTasks.TimeForTask()
                IfcScheduleTimeControl.ID()
                IfcScheduleTimeControl.TotalFloat()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, Temp1, Temp2, HighFloat
    DECLARE NUMBER: TotalNumberOfTask = 0, NumberOfHighFloat = 0

    HighFloatValue = HighFloatValue * WorkingHours * 3600

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks

    FOR EACH IfcScheduleTimeControl.ID
        IF(IfcScheduleTimeControl.TotalFloat > HighFloatValue)
            ADD IfcScheduleTimeControl.ID IN Temp1
        END IF
    END FOR
```

```
    FOR EACH ID IN Temp1
        FOR EACH IfcRelAssignsTasks.TimeForTask
            IF (IfcRelAssignsTasks.TimeForTask[i] == Temp1)
                ADD IfcRelAssignsTasks.RelatedObjects[i] to Temp2
            END IF
        END FOR
    END FOR

    FOR EACH ID IN Temp2
        FOR EACH ID IN TotalTasks
        IF (TotalTasks[ID] == Temp2[ID])
            ADD one to NumberOfHighFloat
            ADD ID IN HighFloat
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfHighFloat, Denominator:
    TotalNumberOfTask, value: AllowablePercent, Condition: >, MetricName: "TasksWithOutOfRange%=")

    FOR EACH ID IN HighFloat
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] == HighFloat[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

6. Number of activities with Negative total float

Negative Float % = total # of tasks with negative float × 100 / # of Total tasks

```
BEGIN MAIN TasksWithNegativeTotalFloat (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask,
AllowablePercent = 0)

    '''  INPUT: IfcRelAssignsTasks.RelatedObjects()
                IfcRelAssignsTasks.TimeForTask()
                IfcScheduleTimeControl.ID()
                IfcScheduleTimeControl.TotalFloat()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, Temp1, Temp2, NegativeFloat
    DECLARE NUMBER: TotalNumberOfTask = 0, NumberOfNegativeFloat = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks
```

```
    FOR EACH ID IN IfcScheduleTimeControl
        IF(IfcScheduleTimeControl.TotalFloat[i] < 0)
            ADD IfcScheduleTimeControl.ID[i] IN Temp1
        END IF
    END FOR

    FOR EACH ID IN Temp1
        FOR EACH IfcRelAssignsTasks.TimeForTask
            IF (IfcRelAssignsTasks.TimeForTask[i] == Temp1)
                ADD IfcRelAssignsTasks.RelatedObjects[i] to Temp2
            END IF
        END FOR
    END FOR

    FOR EACH ID IN Temp2
        FOR EACH ID IN TotalTasks
        IF (TotalTasks[ID] == Temp2[ID])
            ADD one to NumberOfNegativeFloat
            ADD ID IN NegativeFloat
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfNegativeFloat, Denominator:
    TotalNumberOfTask, value: AllowablePercent, Condition: >, MetricName:
    "TasksWithNegativeTotalFloat%=")

    FOR EACH ID IN NegativeFloat
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] == NegativeFloat[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

7. Number of Activities with High duration

High Duration % = total # of tasks with high Duration × 100 / # of Total tasks

```
BEGIN MAIN TasksWithHighDuration (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask,
HighDurationValue = 44, AllowablePercent = 5)

    '''  INPUT: IfcRelAssignsTasks.RelatedObjects()
                IfcRelAssignsTasks.TimeForTask()
                IfcScheduleTimeControl.ID()
                IfcScheduleTimeControl.TotalFloat()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''
```

```
    DECLARE LIST: TotalTasks, Temp1, Temp2, HighDuration
    DECLARE NUMBER: TotalNumberOfTask = 0, NumberOfHighDuration = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks

    HighDurationValue = HighDurationValue * WorkingHours * 3600

    FOR EACH ID IN IfcScheduleTimeControl
        IF(IfcScheduleTimeControl.ScheduleDuration[i] > HighDurationValue)
            ADD IfcScheduleTimeControl.ID[i] IN Temp1
        END IF
    END FOR

    FOR EACH ID IN Temp1
        FOR EACH IfcRelAssignsTasks.TimeForTask
            IF (IfcRelAssignsTasks.TimeForTask[i] == Temp1)
                ADD IfcRelAssignsTasks.RelatedObjects[i] to Temp2
            END IF
        END FOR
    END FOR

    FOR EACH ID IN Temp2
        FOR EACH ID IN TotalTasks
        IF (TotalTasks[ID] == Temp2[ID])
            ADD one to NumberOfHighDuration
            ADD ID IN HighDuration
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfHighDuration, Denominator:
    TotalNumberOfTask, value: AllowablePercent, Condition: >, MetricName: "TasksWithHighDuration%=")

    FOR EACH ID IN HighDuration
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] == HighDuration[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

8. Logic Density

Relationships per activity or Relationship ratio or Logic Density = # of Logic links / # Total Tasks

```
BEGIN MAIN LogicDensity (IfcRelSequence, AllowableValue = 2.1 )
```

```
    '''   INPUT: IfcRelSequence.ID()

    DECLARE LIST: TotalTasks
    DECLARE NUMBER: TotalNumberOfTask = 0, LogicLinks = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks
    TotalNumberOfTask = 100 * TotalNumberOfTask

    LogicLinks = COUNT ID in IfcRelSequence

    PRINT CALL FUNCTION MeasureCheck(Numerator: LogicLinks, Denominator: TotalNumberOfTask, value:
    AllowableValue, Condition: >, MetricName: "LogicDensity%=")
END
```

9. Number of Milestone Tasks

There should be minimum 2 milestone

```
BEGIN MAIN MilestoneTasks
    '''   INPUT: IfcTask.IsMilestone()'''

    DECLARE NUMBER: NoOfMilestoneTask = 0

    FOR items in IfcTask
        if(IfcTask.IsMilestone == TRUE)
            ADD one to NoOfTask
        END IF

        IF(NoOfTask < 2)
            OUTPUT "Measure Not Satisfied. Total No. Of MilestoneTask: " + entity;
        ELSE
            OUTPUT "Measure Satisfied.  Total No Of MilestoneTask: " + entity;
        END IF
    END FOR
END
```

10. Schedule Criticality Rate

Schedule Criticality Rate = total # of critical tasks × 100 / # of Total tasks

```
BEGIN MAIN ScheduleCriticalityRate (AllowableValue = 25)

    DECLARE LIST: TotalTasks, CriticalTasks
    DECLARE NUMBER: TotalNumberOfTask = 0, NumberOfCriticalTask = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks
```

```
    CriticalTasks = CALL FUNCTION ScheduleCriticalityRate(IfcRelAssignsTasks, IfcScheduleTimeControl,
IfcTask)
    NumberOfCriticalTask = COUNT ID in CriticalTasks

    PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfCriticalTask, Denominator:
    TotalNumberOfTask, value: AllowableValue, Condition: >, MetricName: "ScheudleCriticalityRate%=")
END
```

11. Near Criticality Rate

Near criticality rate = total # of near critical tasks × 100 / # of Total tasks

```
BEGIN MAIN NearCriticalityRate(IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask,
NearCriticalityFloatValue = 10, CriticalityRate = 10)

    '''   INPUT: IfcRelAssignsTasks.RelatedObjects()
                 IfcRelAssignsTasks.TimeForTask()
                 IfcScheduleTimeControl.ID()
                 IfcScheduleTimeControl.IsCritical()
                 IfcScheduleTimeControl.TotalFloat()
                 IfcTask.ID()
                 IfcTask.Name()
                 IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, Temp1, Temp2, NearCriticalTask
    DECLARE NUMBER: TotalNumberOfTask = 0, NearCriticalTaskCounter = 0

    NearCriticalityFloatValue = NearCriticalityFloatValue *WorkingHours * 3600

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask, IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks

    FOR EACH ID IN IfcScheduleTimeControl
        IF (IfcScheduleTimeControl.IsCritical[i] == TRUE)
            IF(IfcScheduleTimeControl.Totalfloat[i] > 0 && IfcScheduleTimeControl.Totalfloat[i] <
            NearCriticalityFloatValue)
                ADD IfcScheduleTimeControl.ID[i] IN Temp1
            END IF
        END IF
    END FOR

    FOR EACH ID IN Temp1
        FOR EACH IfcRelAssignsTasks.TimeForTask
            IF (IfcRelAssignsTasks.TimeForTask == Temp1)
                ADD IfcRelAssignsTasks.RelatedObjects to Temp2
            END IF
        END FOR
    END FOR
```

```
    FOR EACH ID IN Temp2
        FOR EACH ID IN TotalTasks
        IF (TotalTasks[ID] == Temp2[ID])
            ADD ID to NearCriticalTask
            ADD one to NearCriticalTaskCounter
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: NearCriticalTaskCounter, Denominator:
    TotalNumberOfTask, value: CriticalityRate, Condition: >, MetricName: "NearCriticalityRate%=")

    FOR EACH ID IN NearCriticalTask
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] = NearCriticalTask[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

12. Nos of critical activities with High duration

Critical activities' duration = Maximum 30 days

```
BEGIN MAIN HighDurationCriticalTask(IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask,
HighDurationCriticalTaskValue = 30)

    '''  INPUT: IfcRelAssignsTasks.RelatedObjects()
                IfcRelAssignsTasks.TimeForTask()
                IfcScheduleTimeControl.ID()
                IfcScheduleTimeControl.ScheduleDuration()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, Temp1, Temp2, HighDurationCriticalTask
    DECLARE NUMBER: HighDurationCriticalTaskCounter = 0

    HighDurationCriticalTaskValue = HighDurationCriticalTaskValue * WorkingHours * 3600

    FOR EACH ID IN IfcScheduleTimeControl
        IF (IfcScheduleTimeControl.ScheduleDuration > HighDurationCriticalTaskValue)
            ADD IfcScheduleTimeControl[ID] IN Temp1
        END IF
    END FOR

    FOR EACH ID IN Temp1
        FOR EACH TimeForTask IN IfcRelAssignsTasks
            IF (TimeForTask == Temp1)
```

```
            ADD RelatedObjects to Temp2
        END IF
    END FOR
END FOR

FOR EACH ID IN Temp2
    FOR EACH ID IN TotalTasks
    IF (TotalTasks[ID] == Temp2[ID])
        ADD ID to HighDurationCriticalTask
        ADD one to HighDurationCriticalTaskCounter
    END IF
END FOR

PRINT CALL FUNCTION MeasureCheck(Numerator: HighDurationCriticalTaskCounter, Denominator: 100,
value: 0, Condition: >, MetricName: "HighDurationCriticalTask%=")

FOR EACH ID IN HighDurationCriticalTask
    FOR EACH ID IN IfcTask
        IF (IfcTask[ID] = HighDurationCriticalTask[ID])
            PRINT IfcTask.Name, IfcTask.TaskID
        END IF
    END FOR
END FOR
END
```

Denominator is kept 100 because we don't need percent here.

13. Number of activities with out-of-range duration

Activity duration should be within a min/max range 5 to 90 days

```
BEGIN MAIN TasksWithOutOfRange (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask, MinDuration = 5,
MaxDuration = 90)

    '''   INPUT: IfcRelAssignsTasks.RelatedObjects()
                 IfcRelAssignsTasks.TimeForTask()
                 IfcScheduleTimeControl.ID()
                 IfcScheduleTimeControl.ScheduleDuration()
                 IfcTask.ID()
                 IfcTask.Name()
                 IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, TEMPLIST [Name, TaskID, ScheduleDuration]
    DECLARE NUMBER: Counter = 0

    MinDuration = MinDuration * WorkingHours * 3600
    MaxDuration = MaxDuration * WorkingHours * 3600

    FOR EACH item IN IfcTask.ID()
        FOR EACH ID IN TotalTasks
            IF(TotalTasks_ID == IfcTask_ID[i])
```

```
                FOR EACH IfcRelAssignsTasks.RelatedObjects
                    IF(TotalTasks.ID == IfcRelAssignsTasks.RelatedObjects[j])
                        FOR EACH IfcScheduleTimeControl.ScheduleDuration
                            IF(IfcRelAssignsTasks.TimeForTask[j] == IfcScheduleTimeControl.ID[k])
                                IF(IfcScheduleTimeControl.ScheduleDuration[k] > MaxDuration or
                                IfcScheduleTimeControl.ID[k] < MinDuration )
                                    ADD IfcTask.Name[i], IfcTask.TaskID[i],
                                    IfcScheduleTimeControl.ScheduleDuration[k] IN TEMPLIST
                                    ADD one to Counter
                                END IF
                        END FOR
                    END IF
                END FOR
            END IF
        END FOR
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: Counter, Denominator: 100, value: 0, Condition: >,
    MetricName: "Number of activities with out-of-range duration=")

    FOR EACH item IN TEMPLIST
        PRINT TEMPLIST.NAME, TEMPLIST.TaskID, TEMPLIST.ScheduleDuration
    END FOR
END
```

14. Number of task with duplicate name

Name of task should be unique

```
BEGIN MAIN Tasksduplicate (IfcTask)

    ''' INPUT: IfcTask.Name() '''

    DECLARE LIST: TotalTasks, X, Y
    DECLARE NUMBER: Counter = 0, NumberOfDuplicateName = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask,IfcRelNests)

    FOR EACH ITEM IN TotalTasks
        FOR EACH ITEM IN IfcTask
            IF (IfcTask[ID] = TotalTasks[ID])
                ADD IfcTask.Name to X
            END IF
        END FOR
    END FOR

    Y = X

    FOR EACH X[Name]
```

```
        FOR EACH Y[Name]
            IF (Y[Name] == X[Name])
                ADD one to Counter
            END IF
        END FOR

        IF (Counter > 1)
            Print X[Name]
            ADD one to NumberOfDuplicateName
        END IF

        Counter = 0
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: NumberOfDuplicateName, Denominator: 100, value:
    0, Condition: >, MetricName: "NumberOfDuplicateName=")
END
```

15. WBS Verification

All tasks should have a Work Breakdown Structure (WBS) identifier assigned.
TotalMissingWBS = Total Missing WBS Tasks / Total Tasks

```
BEGIN MAIN WBSVerification(IfcRelNests, IfcTask)

    '''   INPUT: IfcRelNests.RelatedObjects()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, MissingWBS
    DECLARE NUMBER: TotalMissingWBS = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask,IfcRelNests)

    FOR EACH ID IN TotalTasks
        FOR EACH IfcRelNests.RelatedObjects
            IF (IfcRelNests.RelatedObjects != TotalTasks.ID)
                ADD one to TotalMissingWBS
                ADD IfcRelNests.RelatedObjects to MissingWBS
            END IF
        END FOR
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: TotalMissingWBS, Denominator: 100, value: 0,
    Condition: >, MetricName: "TotalMissingWBS=")

    FOR EACH ID IN MissingWBS
        FOR EACH ID IN IfcTask
```

```
            IF (IfcTask.ID == MissingWBS.ID)
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

16. Out of range activity numbers

Activities should be within a min/max range 50 to 1000 (at least one activity for each 10,000

```
BEGIN MAIN OutOfRangeActivityNumbers (MinActivity = 50, MaxActivity = 1000)

    '''   INPUT: MinActivity, MaTotalTasksActivity '''

    DECLARE LIST: TotalTasks
    DECLARE NUMBER: TotalNumberOfTask = 0

    TotalTasks = CALL FUNCTION TotalTasks(IfcTask,IfcRelNests)
    TotalNumberOfTask = COUNT ID in TotalTasks


    IF (MinActivity < TotalNumberOfTask < MaxActivity )
        PRINT "Measure Satisfied. TotalNumberOfTask:" + TotalNumberOfTask
    ELSE
        PRINT "Measure Not Satisfied. TotalNumberOfTask:" + TotalNumberOfTask
END
```

17. Critical Activity Affiliation

Each critical activity should have a predecessor

```
BEGIN MAIN CriticalActivityAffiliation(IfcRelAssignsTasks, IfcTask)

    '''   INPUT: IfcRelAssignsTasks.RelatedObjects()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: CriticalTasks, Missing
    DECLARE NUMBER: Counter

    CriticalTasks = CALL FUNCTION CriticalTasks (IfcRelAssignsTasks, IfcScheduleTimeControl, IfcTask)

    FOR EACH ID IN CriticalTasks
        IF (CriticalTasks[ID] != IfcRelAssignsTasks.RelatedObjects[ID])
            ADD ID to Missing
            ADD one to Counter = 0
        END IF
```

```
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: Counter, Denominator: 100, value: 0, Condition: >,
    MetricName: "Critical Activities with Missing Predecessor=")

    FOR EACH ID IN Missing
        FOR EACH ID IN IfcTask
            IF (IfcTask.ID == Missing[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

18. Number of Relationship with odd Lag

lag duration = Maximum 5 days

```
BEGIN MAIN NumberOfRelationshipWithOddLag (IfcRelSequence, IfcTask, MaxLagInRelation = 5)

    '''  INPUT: IfcRelSequence.TimeLag()
                IfcRelSequence.RelatingProcess()
                IfcRelSequence.RelatedProcess()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: Temp[RelatingProcess, RelatedProcess]

    MaxLagInRelation = MaxLagInRelation * WorkingHours * 3600

    FOR EACH IfcRelSequence.TimeLag
        IF (IfcRelSequence.TimeLag[i] > MaxLagInRelation)
            ADD RelatingProcess[i], RelatedProcess[i] to Temp
            ADD one to Counter
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: Counter, Denominator: 100, value: 0, Condition: >,
    MetricName: "Number of Relationship with odd Lag=")

    FOR EACH item IN Temp
        FOR EACH ID IN IfcTask
            IF (IfcTask.ID == Temp.RelatingProcess)
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF

            IF (IfcTask.ID == Temp.RelatedProcess)
                PRINT IfcTask.Name, IfcTask.TaskID
```

```
        END IF
    END FOR
  END FOR
END
```

19. SF Relationship Type

SF relationship should be avoided

```
BEGIN MAIN SFRelationshipType (IfcRelSequence, IfcTask, AllowableValue = 0)

  '''  INPUT: IfcRelSequence.SequenceType()
              IfcRelSequence.RelatingProcess()
              IfcRelSequence.RelatedProcess()
              IfcTask.ID()
              IfcTask.Name()
              IfcTask.TaskID()'''

  DECLARE LIST: Temp[RelatingProcess, RelatedProcess]
  DECLARE NUMBER: Counter = 0

  FOR EACH IfcRelSequence.SequenceType
    IF (IfcRelSequence.SequenceType[i] == START_FINISH)
      ADD IfcRelSequence.RelatingProcess[i], IfcRelSequence.RelatedProcess[i] to Temp
      ADD one to Counter
    END IF
  END FOR

  PRINT CALL FUNCTION MeasureCheck(Numerator: Counter, Denominator: 100, value: AllowableValue,
  Condition: >, MetricName: "SF Relationship Type=")

  FOR EACH item IN Temp
    FOR EACH ID IN IfcTask
      IF (IfcTask.ID == Temp.RelatingProcess)
        PRINT IfcTask.Name, IfcTask.TaskID
      END IF

      IF (IfcTask.ID == Temp.RelatedProcess)
        PRINT IfcTask.Name, IfcTask.TaskID
      END IF
    END FOR
  END FOR
END
```

20. Number of Relationship with lags more than successor or predecessor duration

Lags should not be greater than the duration of Predecessor or Successor activity of that relation

```
BEGIN MAIN RelationshipWithMoreLags (IfcRelSequence, IfcRelAssignsTasks, IfcTask)
```

```
    '''  INPUT: IfcRelSequence.ScheduleDuration()
                IfcRelSequence.RelatingProcess()
                IfcRelSequence.RelatedProcess()
                IfcRelAssignsTasks.RelatedObjects()
                IfcRelAssignsTasks.TimeForTask()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: Temp1[RelatingProcess, RelatedProcess, TimeLag, Name, TaskID]

    FOR EACH IfcRelSequence.TimeLag
        IF (IfcRelSequence.TimeLag[i] > 0)
            ADD IfcRelSequence.RelatingProcess[i], IfcRelSequence.RelatedProcess[i],
            IfcRelSequence.TimeLag[i] to Temp1
            FOR EACH ID IN IfcTask
                IF(IfcTask.ID == IfcRelSequence.RelatingProcess[i])
                    ADD IfcTask.Name, IfcTask.TaskID to Temp1
                END IF
            END FOR
        END IF
    END FOR

    PRINT "Tasks whose Relationship have lags more than successor or predecessor duration are following"

    FOR EACH item in Temp1
        FOR EACH item in IfcRelAssignsTasks
            IF(Temp1.RelatingProcess[i] == IfcRelAssignsTasks.RelatedObjects[j])
                IF(IfcRelAssignsTasks.TimeForTask[j] == IfcScheduleTimeControl.ID[k] )
                    IF(IfcScheduleTimeControl.ScheduleDuration[k] < Temp1.TimeLag[i])
                        PRINT Temp1.Name[i], Temp1.TaskID[i]
                    END IF
                END IF
            END IF
            IF(Temp1.RelatedProcess[i] == IfcRelAssignsTasks.RelatedObjects[j])
                IF(IfcRelAssignsTasks.TimeForTask[j] == IfcScheduleTimeControl.ID[k] )
                    IF(IfcScheduleTimeControl.ScheduleDuration[k] < Temp1.TimeLag[i])
                        PRINT Temp1.Name[i], Temp1.TaskID[i]
                    END IF
                END IF
            END IF
        END FOR
    END FOR
END
```

21. Number of activities with high predecessors

An activity should not have more than 10 predecessors

```
BEGIN MAIN HighPredecessors(IfcRelSequence, IfcTask, HighPredecessors = 10, AllowableValue = 0)

    '''  INPUT: IfcRelSequence.RelatedProcess()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, TEMP(ID, Counter = 1)
    DECLARE NUMBER: HighPredecessorsCounter = 0

    HighPredecessors = HighPredecessors * WorkingHours * 3600

    TotalTasks = Call FUNCTION TotalTasks(IfcTask, IfcRelNests)

    FOR EACH Item IN TotalTasks
        FOR EACH Item in IfcRelSequence
            IF (TotalTasks.ID == IfcRelSequence.RelatedProcess)
                FOR EACH ID IN TEMP
                    IF (TEMP[ID] == TotalTasks[ID])
                        Add one to TEMP.Counter
                    ELSEIF(TEMP.ID != TotalTasks.ID)
                        ADD TotalTasks.ID in TEMP.ID
                    END IF
                END FOR
            END IF
        END FOR
    END FOR

    FOR EACH Item in TEMP
        IF(ITEM.Counter > HighPredecessors)
            Add one to HighPredecessorsCounter
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: HighPredecessorsCounter, Denominator: 100, value:
    AllowableValue, Condition: >, MetricName: "Number of Activity with high predecessors=")

    FOR EACH ID IN Temp
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] == Temp[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```

22. Number of activities with high Successors

An activity should not have more than 10 successors

```
BEGIN MAIN HighSuccessors (IfcRelSequence, IfcTask, HighSuccessors = 10, AllowableValue = 0)

    '''  INPUT: IfcRelSequence.RelatedProcess()
                IfcTask.ID()
                IfcTask.Name()
                IfcTask.TaskID()'''

    DECLARE LIST: TotalTasks, TEMP(ID, Counter = 1)
    DECLARE NUMBER: HighSuccessorsCounter = 0

    HighSuccessors = HighSuccessors * WorkingHours * 3600

    TotalTasks = Call FUNCTION TotalTasks(IfcTask, IfcRelNests)

    FOR EACH Item IN TotalTasks
        FOR EACH Item in IfcRelSequence
            IF (TotalTasks.ID == IfcRelSequence.RelatingProcess)
                FOR EACH ID IN TEMP
                    IF (TEMP[ID] == TotalTasks[ID])
                        Add one to TEMP.Counter
                    ELSEIF(TEMP.ID != TotalTasks.ID)
                        ADD TotalTasks.ID in TEMP.ID
                    END IF
                END FOR
            END IF
        END FOR
    END FOR

    FOR EACH Item in TEMP
        IF(ITEM.Counter > HighSuccessors)
            Add one to HighSuccessorsCounter
        END IF
    END FOR

    PRINT CALL FUNCTION MeasureCheck(Numerator: HighSuccessorsCounter, Denominator: 100, value:
    AllowableValue, Condition: >, MetricName: "Number of Activity with High Successors=")

    FOR EACH ID IN Temp
        FOR EACH ID IN IfcTask
            IF (IfcTask[ID] == Temp[ID])
                PRINT IfcTask.Name, IfcTask.TaskID
            END IF
        END FOR
    END FOR
END
```