DEVELOPING A SCHEDULE MODEL FROM A COST MODELER'S PERSPECTIVE

Ben Kwok
Tecolote Research, Inc., El Segundo, California

Daniel Newkirk
Tecolote Research, Inc., El Segundo, California

ABSTRACT

Cost estimators need to utilize and develop estimating methods for project components beyond cost. Schedule is one of those components. The Space Systems Command (SSC)[1] Financial Management Cost organization, known for the development of the Unmanned Space Vehicle Cost Model (USCM)[2], started development of a Schedule Model in 2018. This journey has yielded some exciting new methods, products, and processes. It has also brought challenges as we could not address schedule exactly as we would with cost.

INTRODUCTION

HISTORY

In 2018, the Space Systems Command (SSC) Financial Management and Cost organization began development of a new capability: performing data-driven Schedule Risk Analyses (SRAs) for SSC programs. Our initial efforts focused on developing a standardized approach for summarizing (simplifying) an integrated master schedule (IMS), creating a standard work break down structure (WBS) for mapping IMS tasks, and collecting IMSs and other forms of schedule data. The motivation behind this approach was to institute the rigor utilized in the cost community (i.e. normalized program actuals and methods all associated with a standard WBS). The first years of the effort were devoted to establishing a summarization approach which included data extraction scripts, Excel summarization workbooks, and a standard WBS for satellites.

After a few years the project shifted focus as a function of changing circumstances. One major shift was addressing the needs of cost estimators, who also utilize schedules and milestones in their daily work. The second major shift was to focus on getting data to the schedule analysts quicker versus creating an entire system dedicated to only SRAs. What resulted was an integrated Schedule Model which includes the following content and capabilities:

- Schedule database with over 9k records
- Schedule metrics
- A prototype IMS database with search capability

SCHEDULE RISK ASSESSMENTS

In brief, an SRA is a process of characterizing the risks associated with important tasks that have yet to be completed in an acquisition program, and defining the range of potential final durations for each of those tasks based upon those risks. Using these duration distributions, a Monte Carlo simulation is performed to project the most likely completion date of that program. A data-driven SRA is envisioned as performing analysis using a template schedule based on the original contractor schedule, with the uncertainty distributions for each of the task durations being derived from the data of many different (prior) programs. This would be similar to how univariate distributions are defined for certain WBS elements as a part of the USCM Cost Model. By deriving these distributions directly from prior program data, the schedule analyst is provided a more "unbiased" estimate of the uncertainty around each task, albeit one that can be adjusted based on identified risks as appropriate. This approach also enables a more consistent method of performing an SRA relative to current practices.

The Cost Analyst Perspective on a Schedule Model

This project treads new ground within our community. It targets two distinctly separate domains/users (cost estimators and schedulers). Approaching the development of a schedule model through the perspective of a Cost Analyst (i.e. modeler, estimator, etc.) brings both pros and cons.

Some common best practices from the cost community are identified below:

- Use of data collection templates
- Using techniques such as regressions for estimating
- Collecting and storing historical data
- Using the rigor of a WBS to organize data and methods
- Well established process (e.g. normalization, CER development, etc.)

A simplified process for developing a cost model is described below. There are many cost models that exist in the cost community that follow a similar process:

1. Build a data collection template
2. Collect data aligned to the template
3. Normalize cost to the WBS
4. Collect sufficient data from multiple programs to form models
5. Develop methods by WBS

While it is positive that lessons learned from the Cost Analyst's way of doing things can be leveraged, the process above did not work for a schedule model without significant modifications. One of the biggest reasons is that schedule data set (e.g. IMS) is fundamentally different than cost data set (e.g. accounting data, flex file, 1921, etc.). Cost reports, regardless of fidelity, always sum to a logical total; schedules do not behave this way. Schedule data has more dimensions and parameters associated with each date or duration which make each "data point" more complex than the equivalent "cost data point". Schedules will often drop milestones after they have passed, making collection more challenging whereas cost reports retain all elements from start to finish. There is no single authoritative source that captures every single milestone associated with a given program, schedule data is split across multiple, continuously-changing IMS that span the program life. Although some of these nuances were known at the start, the impact on developing a schedule model was not fully appreciated.

Development Approach

The approach to developing the Schedule Model takes a user first approach. *What are the questions that each group asks? What challenges do they have? What are the known solutions they would typically utilize?* These questions have influenced how we have designed the schedule model both in terms of the data collected and the metrics, regression models, and other estimating tools. Specific examples include:

User Group	Questions	Data Driven Solutions
Cost	• Over what duration should I phase my program? • When will the satellite program launch? • Is the program schedule realistic?	• Estimating relationships • Metrics • Analogous data
Schedule	• What's the duration of this critical path task from a similar program? • What is the statistical uncertainty associated with a given task duration based on actuals for similar tasks on completed programs risk distribution for this task?	• Metrics • Analogous data

In addition to data driven methods there are also processes that have been improved by making them more efficient, consistent, easier to use, effective, etc. Issues associated with collecting schedule data, summarizing schedules, and searching for analogous data are explained in this paper.

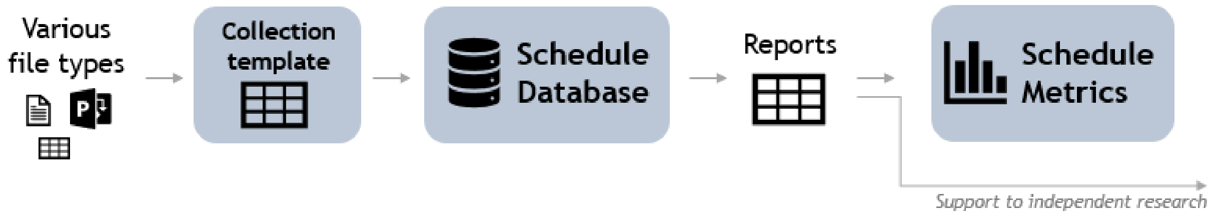
Schedule Model Overview

The Schedule Model consists of three major elements. Each of these elements have some overlap with each other but have distinctly different objectives. The first element is most analogous to a traditional model that involves collection of data to a standard and then building methods from that data; this is referred to as the Schedule Estimating Model. The second element is a schedule task search capability. The third element is a process for creating summarized schedules. The table below describes the different components of the Schedule Model.

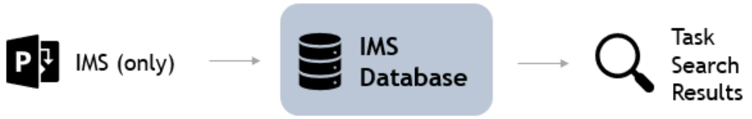
Element	Component	Description
Schedule Estimating Model	Data collection template	Excel template used to capture data; is imported into the Schedule Database.
	Schedule Database	Microsoft Access database that captures select milestone and other meta data. Provides reporting to develop schedule metrics and support other independent research efforts.
	Methods	Comprises of schedule metrics built entirely by data collected as part of the Schedule Model effort.
Schedule Summarization Framework	Python scripts	Used to extract data from an IMS and perform preprocessing steps.
	Excel mapping template	Excel workbook used to map tasks to a standard.
	Schedule-centric standard WBS	Pre-defined standard "schedule WBS" for development and production satellite programs.
Schedule Task Search	IMS Database	Prototype database built using Python and Django. Utilizes IMS extraction Python scripts and Natural Language Processing to enable task name searching.

The diagrams below describe the different components of the Schedule Model.

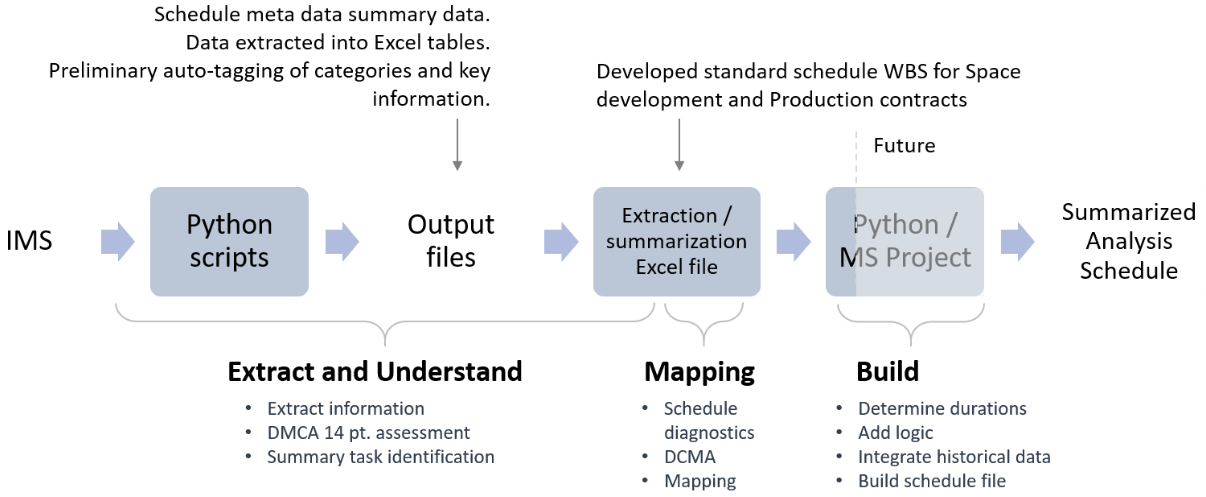
Schedule Estimating Model



Schedule Task Search Capability



Schedule Summarization Model framework


Schedule meta data summary data.
Data extracted into Excel tables.
Preliminary auto-tagging of categories and key information.
Developed standard schedule WBS for Space development and Production contracts
Future
IMS
Python scripts
Output files
Extraction / summarization Excel file
Python / MS Project
Summarized Analysis Schedule
Extract and Understand
Extract information
DMCA 14 pt. assessment
Summary task identification
Mapping
Schedule diagnostics
DCMA
Mapping
Build
Determine durations
Add logic
Integrate historical data
Build schedule file

Each element of the Schedule Model benefits cost estimators and schedulers in different and similar ways. These benefits are likely to change and expand over time.

Elements of the Model	End Users	Benefits
Schedule Estimating Model	Cost estimators	• Data drive methods used to phase cost estimates • Data drive uncertainty for schedule inputs/assumptions in cost estimates
	Schedulers	• Data driven uncertainty bounds in SRAs
Schedule Task Search	Cost estimators	• Faster data collection to develop Schedule Estimating Relationships
	Schedulers	• Access to historical analogous tasks similar to tasks on the critical path (for SRAs)

Elements of the Model	End Users	Benefits
Schedule Summarization Framework	Schedulers	• Consistent means of summarizing an IMS (for SRAs)

Schedule Estimating Model

The Schedule Estimating Model contains 3 components discussed in this paper.

- Data collection process
- Schedule Database
- Schedule metrics and related schedule research

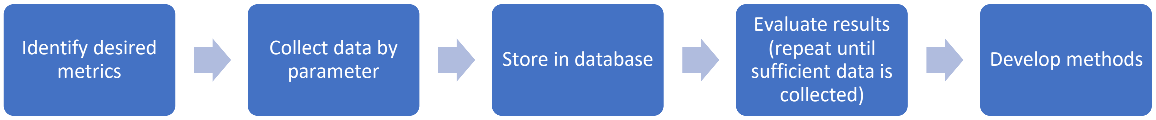
Data Collection Process

Collecting data for the Schedule Model is arguably the most challenging component. This section discusses issues we have encountered along the way.

- Process for identifying what schedule parameters to collect
- Data collection template

Process

Identifying what parameters to collect starts by determining which metrics are desired and targeting data to support them specifically. This provides a higher probability that enough data will be collected to develop the associated schedule methods.


Identify desired metrics
Collect data by parameter
Store in database
Evaluate results (repeat until sufficient data is collected)
Develop methods

Collection Template

The Schedule Model has a unique data collection template to collect a specified set of parameters. Import headers are described in the table below.

Field name	Definition
Program_Name	Program name as defined in the contract. Use parenthesis around vehicle numbers when there is more than 1 flight.
SV_Flight_number	Identify the actual flight number for the vehicle. For technical fields this may include more than one launch of a given satellite design (use commas to separate or dash).
Sequence_name	Applies to software releases or equivalent. Specify the name the program uses. E.g. SW candidate release 3a
Sequence_number	Applies to software releases or equivalent. Specify the numerical order. Example: For Software release 1, 2, 3a, 3b… 3b would be #4.
Type	Categorical field used to distinguish various data types. E.g. milestone, technical, etc.
Type description	Pre-defined name for parameters. E.g. Launch, Contract ATP, etc.
value	Value corresponding with type description (i.e. data point)
Original_name	Original name of parameter as identified in the source document.
Date Type (B/F/A)	Specify if the date is a Baseline, Forecast, or Actual

Field name	Definition
Date Accuracy (P/E)	Specify if the date is a precise value (i.e. the actual date is identified in the document) or estimated (i.e. you had to approximate the date based on a poorly drawn Gantt chart)
Source document	Name of the document or source where the value was captured from
Date of source	Specify the date of the document you captured the value from
Notes	Capture any useful notes here.

Included in the collection template is a definitions list that contains over 130 definitions that span a number of different categories.

- Headers
- General meta data
- Technical meta data
- Standard Satellite milestones
- Advanced Satellite milestones (i.e. lower level milestones)
- Ground system milestones
- Duration calculations

Additionally, examples of completed data collection templates are included in the template.

Lessons learned

Data sourcing needs to be very rigorous. To build cost models, having a single field to identify where a value came from is a standard practice. Because of the complexities of schedule data, multiple aspects of a milestone must be captured to understand and utilize what the milestone represents. As we began this effort we would identify prior schedule related studies (all done by Cost Analysts) that captured milestones and the source in which the date came from. We also notice that data sets would conflict with one another and it became difficult to verify which was more authoritative and correct. This can result in a repeating loop through the data collection process without firm resolution on the best source. To get out of this loop, the data collection template captures additional fields to provide the analyst more confidence in what the milestone date or value represents and where it originated from.

Have a dedicated spreadsheet for capturing data. In prior schedule collection efforts, we've observed that the spreadsheet used to capture milestones is often the same spreadsheet used to analyze the data (i.e. we collect the data in the form we want it to look like for analysis). This approach becomes unmanageable when you want to capture more fields and provide more information about the parameter itself. Also consider that depending on the type of analysis desired, a different data structure may be required.

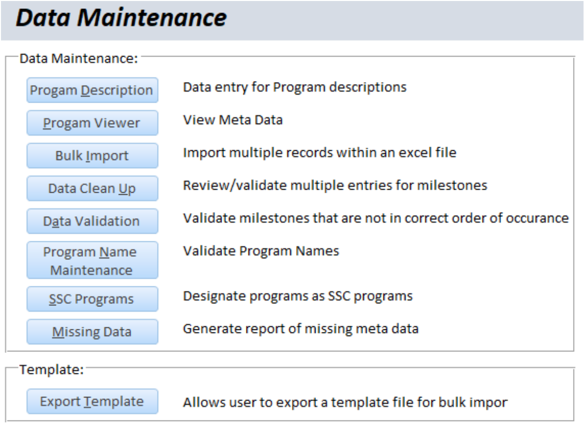
Collect data based on the desired analysis, not by program or WBS. Systematically collecting schedule data is extremely time consuming, particularly if you are trying to normalize that data to common schedule or structure. When we collect data for USCM, we would collect one data set at the end of a program, normalize it to a WBS, and then move to the next program. For schedule, there is no one complete IMS; each IMS is a snapshot of a point in time, and combining multiple schedules is required to achieve a "complete" set of data for a given program. The process of finding and then combining multiple IMSs to yield a complete dataset is a considerable challenge. On an early data collection effort, we spent months of effort getting a single program IMS "right" which increased accuracy but likely captured a lot of detail with unproven value.

Also different from cost data, the individual dates for tasks and milestones can be as important as the normalized data itself, so both would need to be captured depending on the goal of the analysis. For instance, when discussing metrics, we might be interested in how much the total schedule duration increases over time. To perform this analysis, we would require the original and final dates for the start and end of a set of programs at a minimum. Normalized data, while useful for understanding the duration uncertainty around a specific set of tasks, would not be useful to answer this question. In essence, one must decide what analysis is required and gather data for that purpose. Importantly, the way the data is captured and stored in the database doesn't change. What changes is the scope of the data collection effort from "everything" to a targeted subset that can grow over time.

Schedule Database

The Schedule Database is a Microsoft Access database that stores data captured from the Schedule Model data collection template. The database is an analyst tool used to store and maintain the data (i.e. it is not an external user facing tool). Our database also includes a growing list of data maintenance functionality and reporting capability. Additionally, it keeps track of all changes and stores the latest copy of the data collection template.

The maintenance section of the database allows viewing, editing, and uploading of data. There is capability to perform bulk upload of new data entries (via the data collection template) or to simply change or add a value directly. There is a validation section that identifies errors based on a pre-defined ruleset (e.g. preliminary design review occurs after authority to proceed). Maintenance Section Screenshots are shown below:



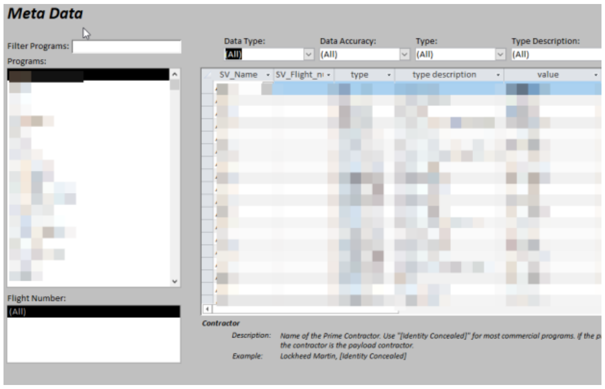
Data Maintenance

Data Maintenance:

Progam Description	Data entry for Program descriptions
Progam Viewer	View Meta Data
Bulk Import	Import multiple records within an excel file
Data Clean Up	Review/validate multiple entries for milestones
Data Validation	Validate milestones that are not in correct order of occurance
Program Name Maintenance	Validate Program Names
SSC Programs	Designate programs as SSC programs
Missing Data	Generate report of missing meta data

Template:

Export Template	Allows user to export a template file for bulk impor



Meta Data

Filter Programs:

Programs:

[redacted]

Data Type: (All)

Data Accuracy: (All)

Type: (All)

Type Description: (All)

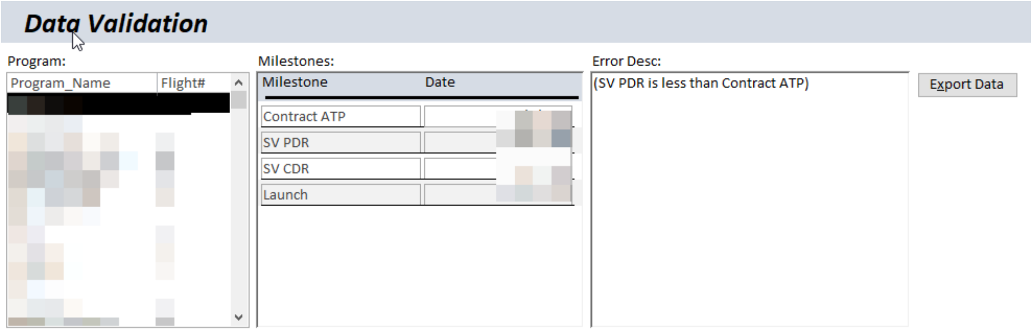
Flight Number:

(All)

Contractor

Description: Name of the Prime Contractor. Use "[Identity Concealed]" for most commercial programs. If the p
the contractor is the payload contractor.

Example: Lockheed Martin, [Identity Concealed]



Data Validation

Program:

[redacted]

Milestones:

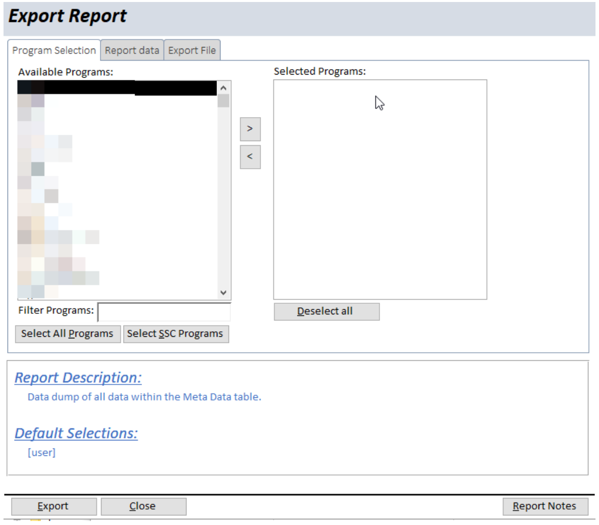
Milestone	Date
Contract ATP	
SV PDR	
SV CDR	
Launch	

Error Desc:

(SV PDR is less than Contract ATP)

The reporting section allows users to explore a predefined list of reports. User has options to pre-filter results by program or add/remove fields from the report prior to export.

Reporting Capability Screenshots



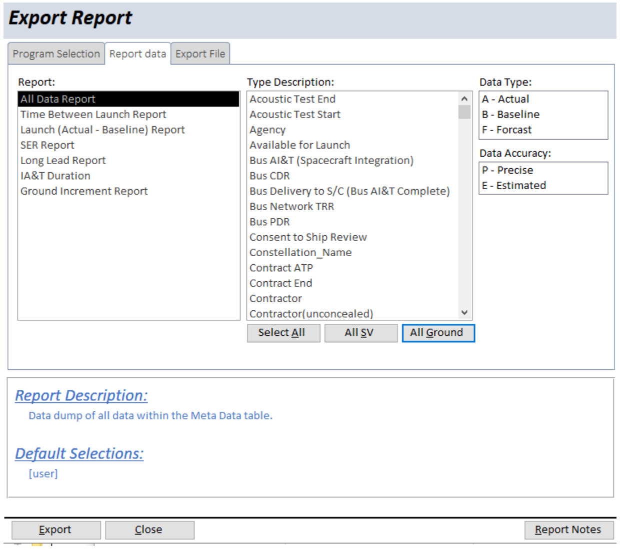
Available Programs:
[redacted]

Selected Programs:

Filter Programs:

Report Description:
Data dump of all data within the Meta Data table.

Default Selections:
[user]



Report:
All Data Report
Time Between Launch Report
Launch (Actual - Baseline) Report
SER Report
Long Lead Report
IA&T Duration
Ground Increment Report

Type Description:
Acoustic Test End
Acoustic Test Start
Agency
Available for Launch
Bus AI&T (Spacecraft Integration)
Bus CDR
Bus Delivery to S/C (Bus AI&T Complete)
Bus Network TRR
Bus PDR
Consent to Ship Review
Constellation_Name
Contract ATP
Contract End
Contractor
Contractor(unconcealed)

Data Type:
A - Actual
B - Baseline
F - Forcast

Data Accuracy:
P - Precise
E - Estimated

Report Description:
Data dump of all data within the Meta Data table.

Default Selections:
[user]

Lessons Learned

- **Excel was not a proper data storage and reporting tool for this application**. Data storage via Excel was insufficient given the functionality desired.
- **Reporting structure is hard to get right on the first try.** Don't spend too much time on the initial report structure and incorporate a feedback mechanism in the report development process.
- **Data needs to be maintained over time.** The Schedule Database has built-in data validation functionality to help identify holes in the data and inconsistent entries.

Schedule Metrics

As of this paper, three schedule metrics have been developed with a series of other metrics in the works. Metrics are quantitative measurements used to track the performance of specific business processes at an operational and tactical level.[3] The table below provides a status of the different metrics in development. In many cases, the motivation for developing the metric was driven by the fact that none currently existed or that the current scheduling approach was based on a rule of thumb.

Metric	Domain	Status
Duration between Long Lead and Production award	Space	Developed
Launch growth from baseline to actual	Space	Developed
Time between sequential launches	Space	Developed
Time between software increments	Ground/SW	In progress
Time between software deliveries	Ground/SW	In progress
Thermal Vacuum (TVAC) related durations	Space	Idea phase
IA&T duration	Space	Idea phase
Time to Initial Launch Capability (ILC)or time between ILC and actual launch	Space	Idea phase

Challenges and Lessons Learned

- **The IMS may not have the milestones required.** Baseline launch dates were found in Special Acquisition Reports (SARs) and other sources, not the contractor IMS. An IMS is not always created for smaller contracts preceding the primary contract.
- **Incorporating "analysis sprints" allowed us to go fast.** We set a two-week maximum sprint duration to quickly provide results back to the team on scope of the effort and to determine if a given analysis or metric should be pursued.
- **Source documents can actually be wrong.** Cost reporting is typically generated from accounting systems with automated validation and crosschecks. Schedule dates are more dependent on manual inputs, engineering judgment, and are more subject to human error and bias. Although rare, some of the dates identified in official reports were clearly wrong.
- **Explanations for why a program schedule slips from its original baseline are complicated.** This is currently out of the scope of this effort.

Other Schedule Estimating Methods

The Schedule Database is also utilized to support other research efforts that require schedule data. The primary example is the development of Schedule Estimating Relationships (SERs), a common estimating methodology used by SSC cost estimators. The Schedule Database also stores the dataset used for developing SSC's SERs. Since the total duration of a program has a significant impact on the overall cost of a program, being able to correctly assess the final duration of that program is critical to producing an accurate cost estimate. Moreover, as the program is executed, various issues may arise that can introduce delays. In fact, the GAO Schedule Assessment guide states that "A cost estimate cannot be considered credible if it does not account for the cost effects of schedule slippage."[4] It therefore becomes important for the cost analyst to have tools to continually assess (and crosscheck) the durations assigned to elements of a schedule.

Traditionally, SERs have been constructed using technical parameters as the independent variables. This enables the analyst to estimate the duration (time to first launch in the case of satellite systems) based on factors such as the technical characteristics of the system being built, the desired lifespan of the system, and complexity factors. However, this approach limits the utility of the SER to the initial portion of a program's lifecycle. Once technical parameters become static, the SER will continue to predict the same duration for the program regardless of any known, existing program delays. As an independent research effort—somewhat separate from the Schedule Model—we explored the use of cost as the independent variable to estimate duration from award to launch. An important finding early on in the study was that the total cost and time to first launch (TT1L) of military space systems were highly correlated. This strong relationship enabled us to develop several SERs using cost data as the independent variable, including for WBS elements such as Space Vehicle, Bus, and Space Vehicle Integration & Test. Beyond providing an additional means of crosschecking the projected TT1L of the program, they also provide another independent method to evaluate the program duration throughout the program lifecycle.

Schedule Summarization Framework

What is Summarization?

Summarization of a schedule is the process of taking a highly detailed IMS and collapsing it into a smaller and more useable size. This is oftentimes done (to vary degrees) to create an "analysis schedule" for an SRA or Joint Cost and Schedule Level analysis (JCL)[5]. Beyond decreasing the size of a schedule, another important aspect is combining the tasks in an IMS in a consistent fashion such that the resulting schedule enables an apples-to-apples comparison of the "summarized task" durations across programs. The Schedule Summarization Framework addresses both.

The closest analog to summarization in the cost estimating world is "normalization". Summarization and normalization may be used interchangeably in this paper as a means of establishing familiarity to the reader. These terms are not actually the same however, as the purpose of normalization is to make data sets consistent with one another. In this case, the methods described below both adjust the resulting schedule size and make the schedule consistent with other data collected.

WHY NORMALIZE A SCHEDULE

What benefit is there to generating a normalized schedule for a program? The benefits are similar to those of generating normalized cost data—building models that facilitate predictions as well as an understanding of the uncertainty around such predictions. These resulting Cost Estimating Relationships (CERs) are then plugged into larger cost models to assess the total cost of a program. In the scheduling world, one "cost model" equivalent would be the SRA or JCL. Both of these techniques are used to predict the total duration of a program, and are typically performed when issues arise during program execution. Monte Carlo simulation is used just as in a traditional cost model, with uncertainty distributions defined for each segment of the schedule under review.

These uncertainty distributions are a critical piece of how well an SRA or JCL will predict a probable date of completion. Schedule Risk Assessments commonly use a triangular distribution, parameterized by a minimum, most likely and maximum duration.[6] In a conventional SRA process, those durations are based on subjective inputs of SMEs, injecting potential bias into the results. In contrast to CERs, the lack of normalized schedule data precludes the use of data-derived uncertainty distributions and parameters. The only means of providing direct data to inform tasks/segment uncertainty is to identify analogous tasks from similar programs. Analogous tasks can be difficult to identify and may not always exist (more on this topic in the next section).

In an idealized world, a method for normalizing schedule data and fitting distributions to the duration uncertainty for each normalized task would enable the best means of producing a truly data-driven approach to schedule risk analysis. However, a robust means of quickly and consistently normalizing a schedule is necessary for both the data collection step and the construction of the analysis schedule used in the analysis, as the schedule used in an SRA or JCL itself must be normalized accordingly for those distributions to hold true.

HOW TO SUMMARIZE A SCHEDULE

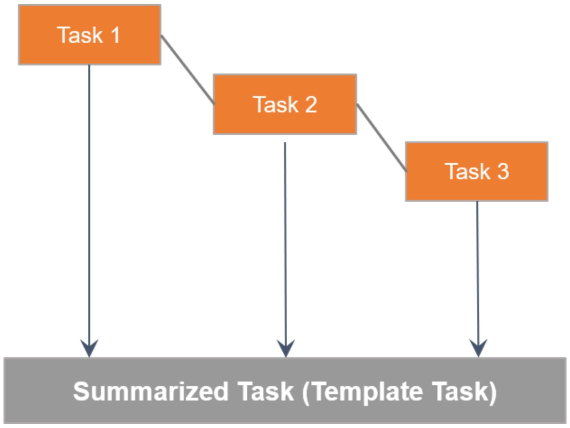
A primary motivation of building a detailed schedule normalization approach then becomes defining the probability distributions that describe the durations of different portions of the program lifecycle. A normalized schedule is commonly described as a "summarized" schedule, where an integrated master schedule is used as input, a process of consistently combining tasks is performed, and the output is a collection of "summarized tasks" that enable apples-to-apples comparisons and appropriate analysis and model building efforts. In the case of a cost model, we would map our costs to a WBS. In the case of a schedule or IMS, we would need to map our schedule to a template schedule or WBS equivalent. Effectively, the top-level process is the same as with cost data, while the actual implementation is very different.

The required components of summarization include:

1. A pre-defined schedule template or structure
2. A method for consistently mapping tasks in an IMS to each portion of the schedule template
3. A method for summarizing or simplifying the schedule logic

A predefined schedule template lists out the different tasks available, which are used to build a summarized schedule. The analyst (or software) will choose those template tasks that are relevant to a current program and with the required level of granularity. Once the set of tasks are chosen, the detailed tasks of an IMS are then "mapped" to the template tasks (see figure below). This means that the analyst (or software) will identify the activities in the detailed task from an IMS and find the proper template task that includes those activities. Once the detailed tasks are "mapped" to the template task, the final duration of the template task are determined. After this step, the logic of the mapped tasks are merged/simplified to reflect an updated schedule logic with appropriate relationships between the template tasks. Since these template tasks are common to all programs, the durations associated with each can be compared across programs, similar to how normalized cost data can be compared across programs for a given WBS level.

Mapping Tasks to a Template



To date, SSC has created several templates, including pre-defined schedule templates for satellite programs and ground programs. Additionally, several programs have had their schedules manually mapped to these schedule templates. The rules behind how tasks are mapped have undergone review by schedule SMEs and are continually being refined. SSC is leveraging these manually-summarized schedules to facilitate development of machine learning techniques to help speed up the summarization process and improve data collection efforts. We have found that the time limitations of performing SRAs and the large body of programs to be normalized have, long term, necessitated a means of expediting and or automating the normalization process.

AUTOMATING NORMALIZATION

Given the difficulty associated with schedule summarization, we set out to design code that would facilitate portions of or even the entire process of normalization. These have included:

- Python scripts that characterize top-level details about an IMS.
- Python scripts that extract data from an IMS in a consistent fashion.
- Python scripts that attempt to identify key milestones or "boundaries" between segments of a program lifecycle using Natural Language Processing.
- A model-based approach to predicting the correct mappings between tasks from an IMS and a schedule template, described elsewhere. [7]

Much like the overall summarization approach, our team is currently developing the software necessary to automate summarization. Exciting new techniques and software to automate this process are taking shape and we expect this to be a focus of future innovation for the cost and schedule analysis community.

Challenges

- Creating a new normalization process from scratch
- Addressing the normalization of schedule logic
- Developing software to facilitate the normalization process in tandem with developing the process itself

Lessons Learned

- **Compromises on schedule template granularity are necessary to maintain usability.**
- **Time constraints for performing an analysis task are important.** Even the absolute best methodologies can't be used if they can't be accomplished in the time allotted.
- **Software facilitating normalization and data collection is increasingly a necessity.**

Schedule Task Search

Defining the Problem

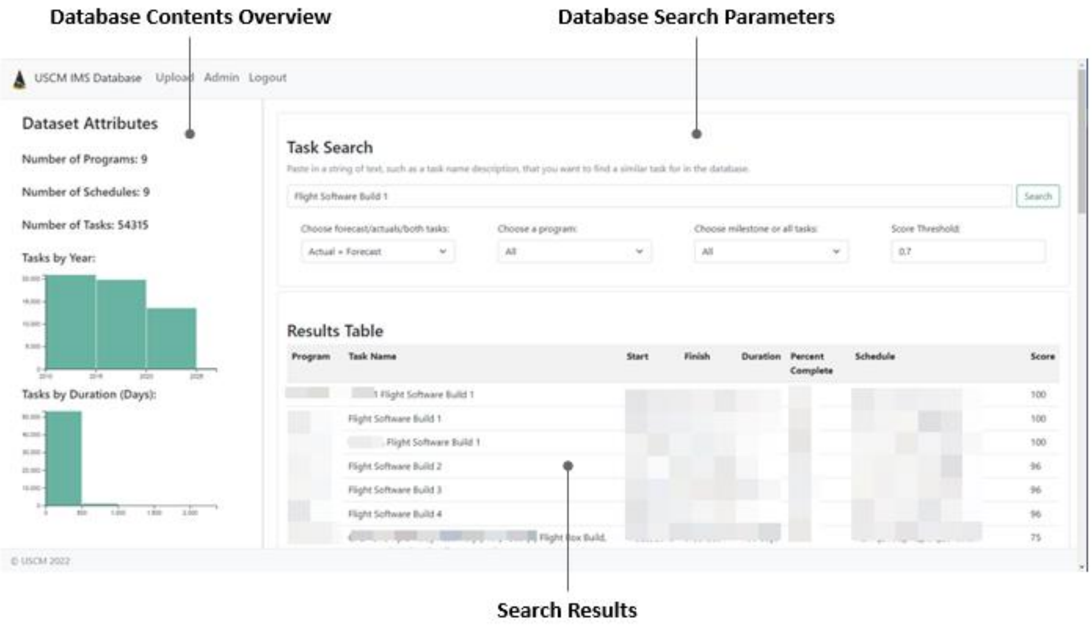
The ability to immediately retrieve schedule task data is beneficial to both cost analysts and schedulers. Discovery of these data points may require digging through several to many IMSs to get sufficient information and a time commitment of days to weeks—entirely dependent on the number of tasks one is searching for of course. While Microsoft Project has built in search functionality, it is unable to identify analogous tasks outright. Our separate Schedule Database is focused on collecting specific dates that reflect key milestone dates. IMS's contain far more data that isn't captured by our data collection efforts.

For schedulers, specifically, performing data driven SRAs requires use of historical analogous data. Our group has provided support in this area on an ad-hoc basis by developing and running Python scripts that use Natural Language Processing (NLP) to compare the description of a series of tasks against other schedules. The natural evolution of this was to improve and proliferate this capability.

A schedule search engine

Over time, Python scripts were refined and yielded highly accurate matches as validated by schedule analysts. The primary limitation of these scripts is the amount of data available to them, as they were essentially designed to compare one set of tasks to another. To address this issue, the algorithms were adjusted to scale to a larger collection of IMSs and were embedded into a web-based application/search engine; internally, we refer to this as the IMS Database. The IMS Database was built using Python and Django (a Python-based web framework) and runs on Windows Server products. A screenshot of the prototype search engine is shown below.

IMS Database Task Search Capability



Dataset Attributes

Number of Programs: 9

Number of Schedules: 9

Number of Tasks: 54315

Tasks by Year:

Tasks by Duration (Days):

Task Search

Paste in a string of text, such as a task name description, that you want to find a similar task for in the database.

Flight Software Build 1

Choose forecast/actuals/both tasks: Actual + Forecast

Choose a program: All

Choose milestone or all tasks: All

Score Threshold: 0.7

Results Table

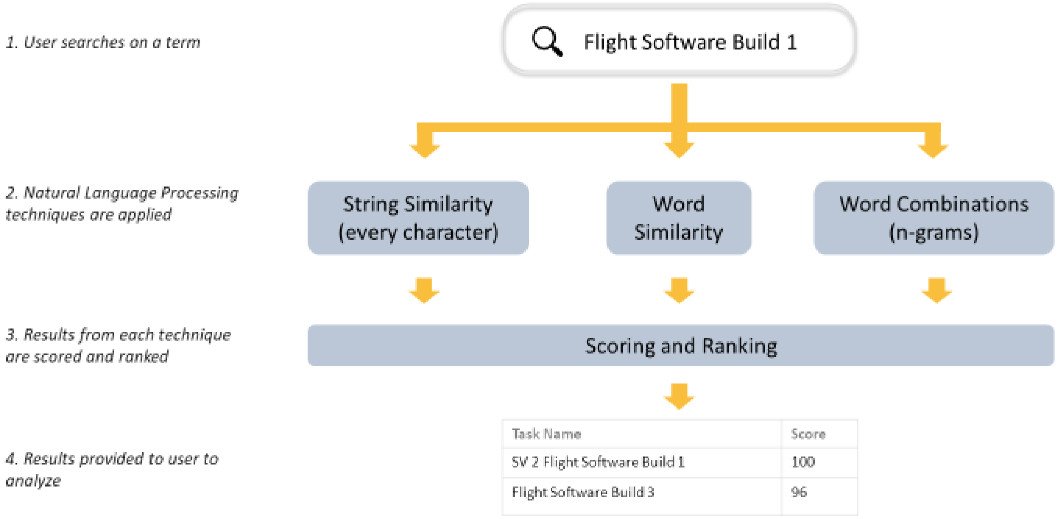
Program	Task Name	Start	Finish	Duration	Percent Complete	Schedule	Score
[redacted]	[redacted] 1 Flight Software Build 1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	100
[redacted]	Flight Software Build 1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	100
[redacted]	[redacted] Flight Software Build 1	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	100
[redacted]	Flight Software Build 2	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	96
[redacted]	Flight Software Build 3	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	96
[redacted]	Flight Software Build 4	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	96
[redacted]	[redacted] Flight [redacted]ox Build,	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	75

Natural Language Processing

Over the course of our research efforts, we have found that an ensemble[8] of different methods are necessary to identify matching tasks. Task descriptions—and occasionally other metadata—are often written in the voice of a schedule analyst. That is, the language used to describe an activity is determined by a specific person even when using best practices. One can even sometimes identify when tasks descriptions were written by two separate individuals. This aspect of an IMS impacts the approach we use to determine a "best match" for a given task (assuming one exists).

In order to incorporate multiple algorithms into a single solution, a ranking system is used. Each algorithm produces a score that is descriptive of the "closeness" or "distance" between any two task descriptions and their associated metadata. The ranking system ensures that the scores are scaled appropriately to be comparable and then "chooses" the best match based on the best rankings; in the event of each algorithm identifying a different task, the method with the highest score is chosen (see figure below). This ranking system is currently a heuristic approach, but is being converted into a machine learning ranking (MLR)[9] approach as sufficient data for learning the appropriate parameters has been gathered. Beyond the current approaches used for matching, the search capability can add additional algorithms to improve accuracy easily given the voting system implementation.

Ranking Matches for Task Search


1. User searches on a term
Flight Software Build 1
2. Natural Language Processing techniques are applied
String Similarity (every character)
Word Similarity
Word Combinations (n-grams)
3. Results from each technique are scored and ranked
Scoring and Ranking
4. Results provided to user to analyze

Task Name	Score
SV 2 Flight Software Build 1	100
Flight Software Build 3	96

Some of the NLP-based methods used behind the scenes include these standard preprocessing techniques:

- Tokenization – breaking strings/sentences into words
- Lemmatization – identifying the root of a word
- Parts of Speech – using a Hidden Markov Model to identify the parts of speech for a word given preceding words
- Synonyms – identifying synonyms for a given word and looking for those synonyms in a potential matching task
- Abbreviation/Acronym identification – identifying if a word is an abbreviation for another word or stands for a set of words.
- Spell Checking – identifying and correcting common spelling errors

Some of the techniques used for matching include:

- Edit Distance[10]
- Percent of common words/synonyms
- Percent of common N-grams[11]

Importantly, the techniques used consider word order, usage of common phrases, and other characteristics of a task/metadata description. Again, since these are likely to have been written by two different people, the language used to describe each will require evaluating many different characteristics (features in ML language) of the strings to identify the best potential match.

FUTURE DEVELOPMENT

The IMS Database is currently a prototype, but has already shown value in enabling schedulers to have an additional, important input into their analysis. It also serves as an invaluable source of data for future research and analysis of schedule data. Areas of current research include:

- Use of Neural Networks (Deep Learning) to facilitate improved matching and scalability.
- Final definitions of the data and metadata to be extracted from and stored for each schedule snapshot.

- Techniques for performing analysis of data across snapshots for a single program.

Lessons Learned

- **Search capabilities are necessary to find relevant data.** The development and use of sophisticated search algorithms are not a commonplace solution in the cost estimating world but have found a home for identifying analogous schedule tasks. The use of search is incredibly efficient when compared to the approach of first mapping tasks to a standard structure.
- **Some algorithms take a long time to compute, and may require clever ways to efficiently manage their use.** Research efforts can identify useful ML or NLP approaches individually, but integrating these together into an efficient tool can be difficult. The computational complexity, memory requirements, and other considerations can require evaluating tradeoffs and alternate approaches to those originally developed. As the body of data grows, finding ways to accurately identify all of the relevant data points in the database in a timely manner takes planning and experimentation.

Conclusion

The Schedule Model is not a typical model. Because it addresses different user bases, it has multiple sections to address different needs. Both cost estimators and schedulers need schedule data but use it differently. Although superficially similar, cost and schedule cannot be treated the same. New processes and approaches need to be developed, but without starting from ground zero. Numerous lessons were learned along the way, which were documented in this paper.

This work was funded by the Space Systems Command, contract GS00Q14OADS619. The authors thank Ms. Adriana Contreras and Mr. Raj Palejwala for their support of this effort.

Notes and references

[1] https://www.ssc.spaceforce.mil/Connect-With-Us/Space-Systems-Command-Front-Door
[2] http://www.uscmonline.com/
[3] https://www.datapine.com/blog/kpis-vs-metrics-differences/#what-are-metrics, accessed January 20th, 2023
[4] GAO, "Schedule Assessment Guide: Best Practices for Project Schedules", p. I (Preface), Released December 2015
[5] https://www.nasa.gov/sites/default/files/files/JCL_Overview_Brochure.pdf
[6] "Joint Agency Cost Schedule Risk and Uncertainty Handbook", p. A-19, Released September 2014
[7] Newkirk, Daniel, *Summarizing Schedules using Hidden Markov Models and Natural Language Processing,* NASA Cost and Schedule Symposium, April 21, 2021 (Virtual)
[8] 1.11. Ensemble methods — scikit-learn 1.2.0 documentation, accessed January 22nd, 2023
[9] https://en.wikipedia.org/wiki/Learning_to_rank, accessed January 23rd, 2023
[10] Brill, Eric; Moore, Robert C. An Improved Error Model for Noisy Channel Spelling Correction. Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, 2000 pp. 286–293.
[11] https://en.wikipedia.org/wiki/N-gram, accessed January 20th, 2023