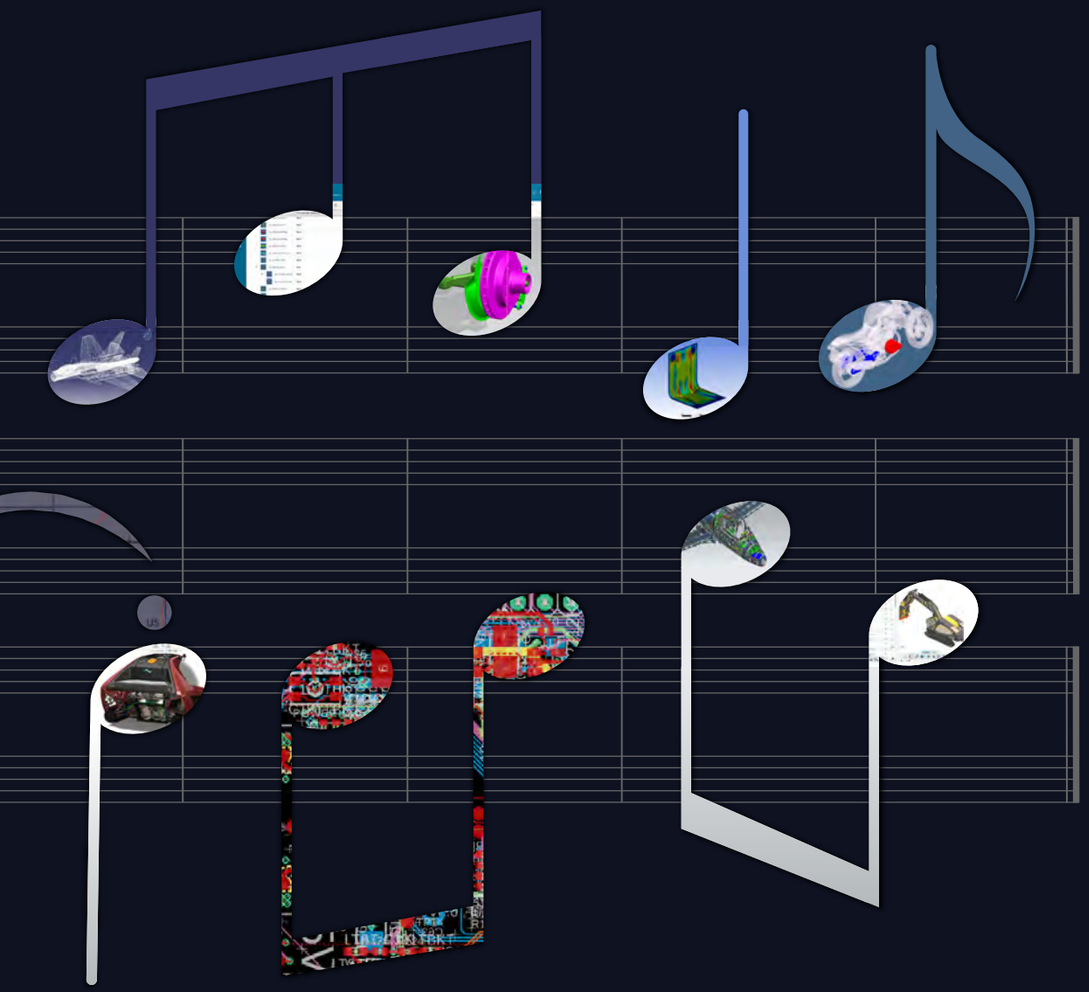
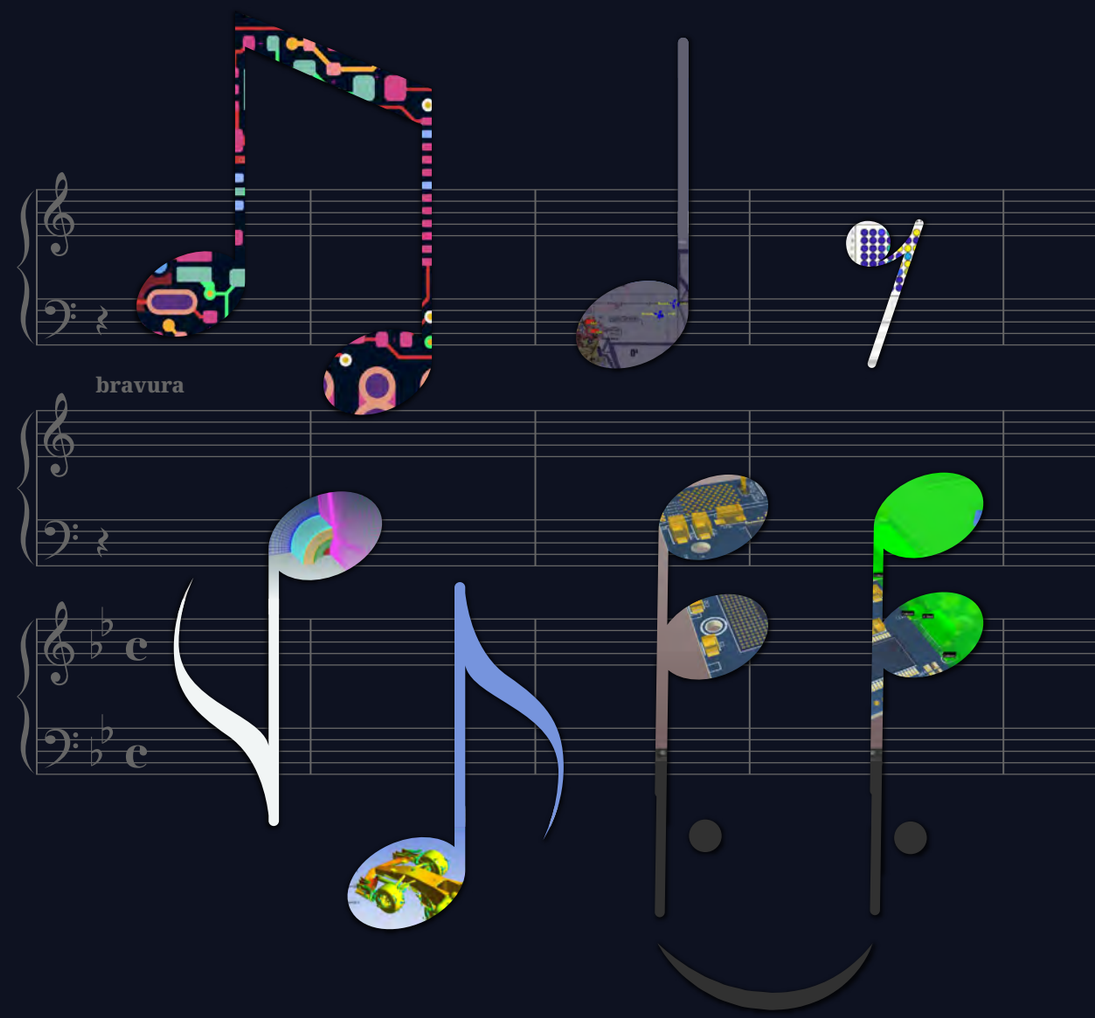


istari
DIGITAL

From Virtuoso to Maestro CREATING THE ENGINEER OF THE FUTURE

From Virtuoso to Maestro

CREATING THE ENGINEER OF THE FUTURE



Written By Istari Digital

Table of Contents

Read This: Preparing for the Future of Digital Engineering

In the next few years, the way engineering is conducted will undergo a seismic transformation. A small group of high-performing organizations will not only adapt—they'll redefine the field of digital engineering, leaving their competitors behind. These pioneers will set new standards for innovation, shrinking design-to-production timelines from years to months, all while achieving unprecedented levels of quality, speed, and cost efficiency.

For organizations that hesitate, the gap will grow insurmountable, impacting everything from market position to profitability. But for those who act decisively, the opportunities will be immense.

THE KEY DRIVERS OF THIS REVOLUTION:

1. SEAMLESS COLLABORATION ACROSS BOUNDARIES

New technologies will dismantle silos, creating ecosystems where teams across disciplines, organizations, and geographies can collaborate like never before.

2. EMPOWERING ENGINEERS WITH INTUITIVE TOOLS

User-friendly, integrated toolsets will enable engineers at every level to experiment, iterate, and refine faster than ever, unlocking their full creative and technical potential.

3. REVOLUTIONIZING MINDSETS AND MENTAL MODELS

Organizations will need to adopt new ways of thinking, encouraging their engineers, managers, and executives to embrace the mindset of the "engineer of the future."

This book outlines the critical pillars your organization must invest in to lead this transformation. Using the analogy of music, we'll explore how digital engineering can evolve in much the same way music has—moving from acoustic simplicity to electrified complexity, from solo performances to symphony orchestras where every instrument harmonizes seamlessly.

The engineering revolution ahead will demand more than just new tools; it will require rethinking how organizations function. Those that succeed will become the maestros of their industries—conducting innovation, growth, and competitive advantage on a scale never seen before.

Why This Matters to You

The choices your organization makes today will determine whether you lead this revolution or struggle to catch up. This book provides a roadmap for positioning your organization to thrive in the future of digital engineering.

1

The Dawn of a New Field of Engineering

Rarely do we get to experience a moment like this: the dawning of an emerging field of engineering that will forever change the way we conceive of, design, refine, create and reify our ideas. We are on the precipice of a tremendous wave of technological convergence which will require new ways of educating, training and developing engineers of the future – in fact, new ways of thinking altogether. Welcome to the era of Digital Engineering—a transformative period where the integration of advanced digital-tools, artificial intelligence, and secure data systems will redefine every aspect of innovation and production of physical creations.

This revolutionary period of time will fundamentally alter how we collaborate across engineering disciplines, across industries, and even borders, creating unprecedented opportunity. Soon, we will have the ability to design, simulate, test, and iterate completely in virtual environments, allowing us to solve complex problems faster and with greater precision than ever before. A single engineer will be able to harness the depths of knowledge of the best engineers that the world has ever known – being able to take their thoughts and feelings and translate them smoothly into creations leveraging thousands of years of engineering wisdom.

The future of engineering is here, and it demands agility, creativity, and a commitment to a growth mindset.

Consider Venice. When the early settlers moved to the marshy islands to escape the Visigoths and Huns, they found themselves on unsuitable ground. It could barely hold a person let alone a building. They invented ways to occupy a very soft marsh by placing wooden piles close to each other, forcing out the water between them, and distributing the weight of the building across them. To this day those wooden piles are still intact more than 1500 years later. The sea water was undrinkable and getting water from the mainland to the islands to quench the needs of the growing city was untenable, so they created unique cisterns that would funnel rainwater from roofs and the squares, filter it through sand and gravel and supply it to a public well in the center of the square, many which still exist today. The early Venetians solved these problems with creativity and knowledge passed through apprenticeships that spanned many disciplines – not by hiring a set of civil engineers, geotechnical engineers, hydraulic engineers, flood and wastewater experts, coastal engineers, material scientists, marine engineers, mechanical engineers, fluid dynamics experts, urban planning professionals and construction engineers. Formal engineering wouldn't be taught until the 16th century, with the practice spreading to the École Polytechnique in France in 1794, MIT in 1865, and Oxford in 1908.

As the problem complexity increased, we started to need the specializations that only those with extremely deep technical knowledge could provide. Even societies renowned for their ingenuity, like the Venetians with their mastery of shipbuilding and trade, would have struggled to achieve the rapid technological progress of the Industrial Revolution without a similar level of focused expertise.

Now we find ourselves able to create things that only 50 years ago were the realm of science fiction. We have high-resolution bendable computer-screens that are only 4.59mm thick, goggles that mix real life with the virtual, ensuring that shadows, texture, shading and lighting looks completely natural, a 2,717 foot high building (that's ½ a mile!) in Dubai, with wind load management and foundation design that's capable of withstanding the region's harsh desert environment, and a 17 mile long underground ring in Switzerland and France that can hold temperatures near absolute zero and accelerate protons close to the speed of light and smash them together to future our understanding of fundamental physics. However, with all of those incredible accomplishments of the human race, the field of engineering finds itself limited in critical ways.

The Problems With Today's Engineering Practices

In today's engineering world, disciplines like mechanical, electrical, civil, and software engineering often operate in silos, and rarely interact fluidly. Each specialist has their own set of tools, methods, and software, and integrating the tools and their output across teams is slow and cumbersome, particularly as the size of the teams or project grows. Engineers don't often share a common language – so translating complex ideas between disciplines without losing important details is not just a challenge, it creates bottlenecks, frustration and sometimes critical failures.

The Top Problems

INTER-ENGINEERING PRACTICE COMMUNICATION

Different engineering teams often use different sets of tools, CAD software, simulation tools, manufacturing software, etc., which rarely communicate well with each other. This means that a civil engineer using one platform can't easily share data with an electrical engineer who is using another, and this leads to tremendous inefficiencies in coordination and iteration in the design process, and eliminates the ability for real-time collaboration.

CUMBERSOME VERIFICATION WORKFLOW

Workflows are cumbersome, requiring engineers to manually check and verify that work products from different areas are correct, fulfill requirements, and are within tolerance. This can introduce slowdowns, increase the number of meetings, and magnify the potential of errors, especially when there are changes in design or scope.

THE NEED FOR PHYSICAL PROTOTYPES

Even with our current technologies and advanced simulations, testing how mechanical, electrical, and structural systems interact in a fully integrated environment still requires developing physical prototypes which are costly, time-consuming, and impossible to rapidly modify once they've been built.

DATA AVAILABILITY/ACCCESS

Many large engineering projects generate massive amounts of data, much of which is trapped in isolated databases and proprietary formats. As a result, engineers can't access data they need, which eliminates the ability to reuse information from past designs across teams and projects. As a result, new designs are often built from scratch, eliminating the hope of leveraging existing solutions, and needlessly extending timeframes and budgets.

DATA DISSEMINATION

Even if all of those problems were addressed, sharing data and digital assets across teams, organizations and countries introduces security risks. Engineers in projects involving sensitive intellectual property, confidential designs, or strict government compliance controls can't collaborate without tremendous manual work, greatly reducing the speed of collaboration and information sharing.

THE IMPACT

All of this makes it impossible for teams to rapidly iterate with feedback loops that help them to quickly achieve high quality results. In the vast majority of projects, the intent of the design is watered down for the expediency of manufacturability simply because the ability to communicate and colla borate is hindered at every handoff.

A Solution: Reimagining The Future Of Engineering

What if we rethink engineering completely? Start from scratch and see where it leads us? There's too much engineering knowledge that's necessary to know, for any single person to hold all the engineering knowledge from all the engineering disciplines in their head (and perhaps little interest for any one person to do this.) It takes too long to become an expert in all the popular technologies and tools in an ever-broadening technology landscape. So is there a metaphor that could help us reimagine how an engineer in the future might think and act?

Consider the discipline of music. Perhaps you've played an instrument or maybe you just enjoy listening to the Beatles, but in any case you're familiar with what it's like when one person

knows how to play an instrument really, really, well. How did they arrive at that level of proficiency? Much like engineers,expert musicians study and train for many years to develop their mind and their technical approach to working with the instrument.

But the best musicians don't just play the notes – they interpret the notes. This is why, at the moment of writing, there are more than 300 recordings of Bach's 1741 creation the Aria with Diverse Variations (commonly called the Goldberg Variations) available for sale created by the world's best pianists and harpsichordists. The piece was originally written to be played on the harpsichord, which is an extremely limited instrument. The musician presses a key at a moment in time and that's it – that's all they control. They can't indicate how loud the note is, or even how long it lasts. And while Bach specified every note and when it's to be struck, there's a tremendous ability for a virtuoso to create nuances in when those notes are played that affect the emotions of the listener. For our purposes let's call this musician the solo-engineer, someone like James Dyson who, in the 1970s came up with the idea of using the centrifugal force to separate dust from air. It took Dyson 5 years and 5,127 prototypes to perfect it. A transformative product that was the result of a subtle shift in how the problem was addressed using known engineering principles.

If we're to expand the musical analogy, let's think about a piano quartet (piano, violin, viola, cell) or a college-level garage band (drums, bass, lead guitar, rhythm guitar/singer). Each member in a musical group knows their instrument well, but they also have the added level of complexity of having to coordinate and synergize with several other people – who each have their own speciality in order to create something that becomes more than simply the sum of the parts. The number of variations that is created when we consider things like amplitude (how loud the singer or the snare drum is), modulation (the precise note that the violin produces), articulation (how the bow of the cello is moved across the string or how the bassist strikes the strings) and all the other elements of a small band is astronomical. Each element (volume, pitch, timing, phrasing, vibrato, and even micro-timing between instruments) can be infinitely adjusted, creating countless possible combinations. What separates Steely Dan from a Steely Dan cover band is their ability to wordlessly communicate with each other, collaboratively craft a recording with layers of subtlety, and understand the other's abilities without actually being able to perform the other's part at the same technical level. For our engineering analogy let's call this the siloed-engineering team. This is a team that not only knows their discipline

well, but can effectively work together to create a unique solution to an old problem. Foldable screens on new smartphones use a "waterdrop hinge", which solved the challenge of folding a screen 200,000+ times without degrading. To accomplish this an engineering team had to leverage advanced materials and a carefully engineered pivot system that balances flexibility, structural integrity, and compactness. These solutions couldn't be accomplished by one engineer, and it couldn't be done – at the same rate – by teams that weren't working together in the same environment with high levels of efficient communication.

If you're tracking the analogy, we can imagine that as the number of musicians increases and becomes a symphony orchestra that needs to play together, it becomes impossible for that group to do so without a single person who brings it all together, in this case a conductor. Consider Leonard Bernstein who wrote many pieces including West Side Story, and conducted that musical as well many symphonies by Beethoven, Mahler and Shostakovich. With such difficulties that include the inherent technical challenge of producing sound effectively with a large group, the interpersonal complexities of 100 musicians, and the business objectives of the production, how was he able to create so many remarkable recordings while others never accomplish anything near that level of success? While he was famous for saying "To achieve great things, two things are needed: a plan and not quite enough time", his remarkable ability to balance multiple roles: as a conductor, composer, musician, and an educator, helped him achieve radical success and impact, and perhaps gives us insight into what the future engineering leader may have to become.

It's important to note that Leonard Bernstein wasn't an expert in all the instruments used in an orchestra. He could play the piano very well, was proficient with the clarinet (and I imagine he could play a mean triangle), however the most important thing was that he had deep knowledge of how all the instruments in the orchestra worked and how different instruments, played together, contribute to the texture and mood of the piece. If we extend our engineering analogy further, this might start to look like a modern complex engineering project that's coordinated by an extremely talented and knowledgeable systems-engineer. Someone who understands the roles of all the engineers, the organizations they work in, and how to get them to work together to achieve a common goal. But a Leonard Bernstein doesn't scale: there was only one of him, and likewise in an engineering context, there are very few people or organizations that can work at this level of complexity for all the reasons we cited above.

In music we did solve this problem: In 1984 Ray Kurzweil released the K-250, the world's first instrument that enabled a single musician to sound like an orchestra. It was an 88-key keyboard that enabled a single person to access and play thousands of recorded samples of pianos, strings, brass, woodwinds, drums, the human voice, and other instruments. The keys could detect velocity and reproduce sounds with incredibly high fidelity. In a famous demonstration a live audience listening to sounds coming from behind a curtain on a stage couldn't determine if it was a real orchestra or a synthesizer. This invention enabled a single person to more effortlessly create a symphonic work, use software tools to record what the composer played, play those sections back while the composer played a new line on top of the old one, and so forth. The composer could quickly try different instruments, different styles, and edit the score digitally, ultimately being able to create a notated score that they could give to musicians to perform live. The invention of this technology helped countless people develop and refine complex and rich musical ideas in the digital space, before bringing it to the analog space, with a level of execution that the world had never before experienced. If we can do this with music, perhaps we can also do it with engineering?

2

The Evolution Toward The Modern Digital Thread

Many of the issues that occur in modern engineering projects happen because of the increased complexity of modern engineering systems, tools, and requirements as well as the competitive pressures of time and the need for greater efficiency and balancing complex trade-offs. Much like the conductor who used the Kurzweil technology before printing out the score for an orchestra, we must create a way to better connect the dots in the engineering disciplines in order to accelerate the ability to ideate, iterate, and solve problems in the digital space before we create the physical analog.

THE BASIC IDEA IS TO:

1. Find a way to connect the dots across the engineering disciplines in a new system, and
2. Ensure that a single person or group is able to use this new system effectively.

What makes this difficult is the same thing that makes it hard to work with increasingly large groups of multidisciplinary contributors involved in any project. From an engineering perspective, the engineer we're talking about needs to have a deep understanding of an engineering discipline, as well as an awareness of all the other relevant engineering disciplines. The engineer of the future needs to be able to align a group of engineers, provide the right input so they can use their tools effectively, then make sense of the output of the tools. To make things more difficult, most of the environments in which the engineers work, and their data, are siloed.

Data is siloed by engineering discipline (e.g., mechanical, electrical, software, etc.), by tool vendor (e.g. Dassault, Siemens, The MathWorks, etc.), by network (e.g., cloud, on-premise, local-machines, etc.), by organizational boundaries (e.g., IP constraints) and by information security level (e.g., unclassified vs classified). Because of all these silos - it is extremely challenging to connect any piece of data to another system or human.

There have been many attempts to solve this problem. As the need to protect data increased due to cybersecurity threats, so did the complexity of sharing data. And as the licenses that governed the use of the engineering tools became increasingly more restrictive, the ability for an engineer to connect these tools together in meaningful ways has diminished.

In the 2000s, software was developed to connect some of these systems together in manual ways. These early digital threads enabled engineers to have some tools communicate with other tools in their organization, but custom solutions are easy to break when any single tool in the network changes, and developing systems like this requires people who are experts across domains and who could develop production-level code – a very hard combination of skills to hire for.

By the 2010s, some companies began offering suites of proprietary interconnected tools for robust digital threading. While these tools worked well together, this approach forced vendor lock-in. This lock-in prevented collaboration between organizations and contractors who often rely on diverse toolsets.

Now in the 2020s, there's renewed interest in using modern programming tools and AI solutions to enable people to write complex digital threads without needing to be experts in coding. But even with this technology, we still have the same issues of security, license compliance, and robustness. Even if we're able to solve these technical problems with new technology, we still need to determine what knowledge the engineer of the future will need to possess, which skills they will need and what metaphors they will use to guide their thinking and their work.

We will need to develop a new class of engineers who not only understand their own discipline extremely well (as we have now), but also appreciate how the other disciplines work in conjunction with each other to form a comprehensive solution. They will then need to be trained for the skills that enable them to link the tools together using advanced digital threads in order to have the whole system of tools and disciplines working as a single entity. And they will need to possess the leadership skills to ensure that there's effective communication between all the relevant stakeholders.

At first this process will happen with humans working with other humans in the loop so they can set up and validate the connections, then eventually this will evolve to a system which is assisted by advanced AI that can learn the typical tasks and present them to humans for confirmation, and finally statistical models derived from the AI, along with AI supervision, may some day become the engineer's right-hand robot. At each of these steps along the way we will be able to imagine new solutions and create breakthrough innovations that, as of now, we can't yet dream of.

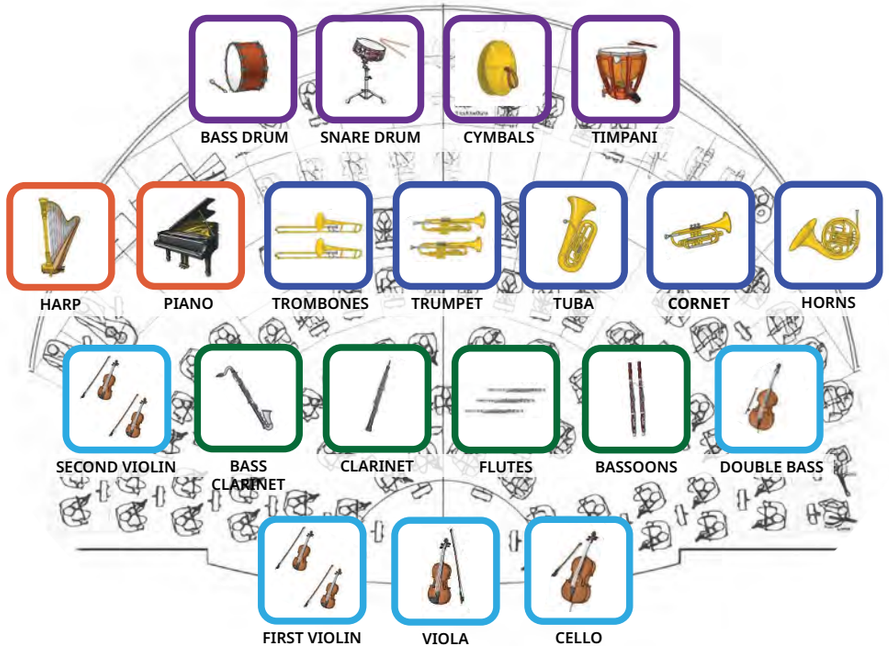
3

Introducing the Orchestra

Much like the instruments in the orchestra, the tools that we use in digital engineering now have specific purposes, strengths, and work particularly well with other tools. There are no famous works for a trio made up of piccolo, snare-drum and accordion, but there are countless ones made for violin, piano and cello. In our digital engineering world, we have many tools that play well with each other, such as Computer Aided Design (CAD), Finite Element Analysis (FEA) and Product Lifecycle Management (PLM) software, as well as tools that require a lot of work to get them to work well together. In this chapter, let us explore the different sections in the orchestra of DE tools, where perhaps the strings are the CAD applications that carry the melody, the brass are the Simulation tools which support the strings, and so forth. The purpose is to enable any engineer to understand how their discipline and the tools they use fit into the context of a new kind of engineering, in order to help them become an effective conductor.

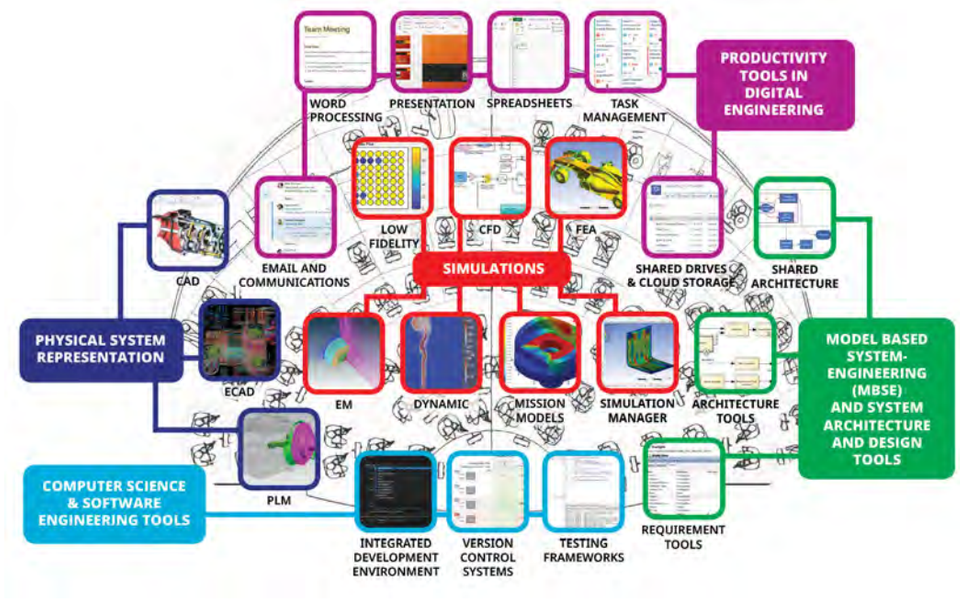
Understanding The Orchestra

The orchestra has evolved over time, adding new instruments, and changing the organization of the sections. A typical symphony orchestra today is often arranged like this:



Here we see how the sections are arranged basically in order of how much sound they produce: strings in the front because the sound is less powerful than brass or percussion and needs to project clearly. Winds and brass in the middle and back because instruments like clarinets, flutes, trumpets, and trombones produce louder sounds, so placing them further back prevents them from overpowering the strings and lastly percussion at the back since percussion instruments are the loudest, and the back position allows their sound to blend naturally with the rest of the orchestra.

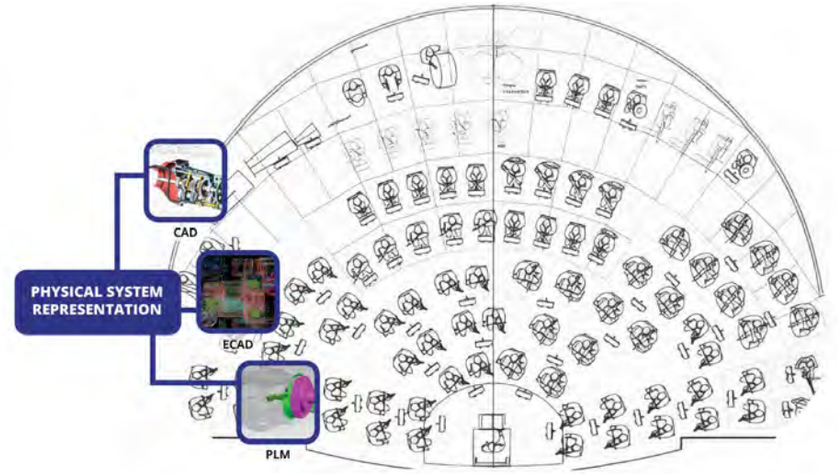
In our digital engineering orchestra, it might look something like this:



The arrangement here doesn't really matter, however we might imagine that most engineering projects rely heavily on CAD drawings (much like how the violins play the melody), and then loop in some analysis tools that support the CAD drawings (like the woodwinds and brass), and everything is working to the timing of the program and project managers, which are like the percussion section, in the back, keeping the beat, (and often the loudest!)

Let build up our orchestra and understand each section a bit better by understanding the different groupings (like Physical System Representations), the sub groupings (in this case, CAD, ECAD and PLM) and common tools for those groupings (PTC Creo Parametric, KiCad and 3DEXPERIENCE CLOUD PLM, for example).

Physical System Representation (e.g. 3D)



Every object we can hold or touch, from the low technology solutions; forks and shoelaces, to the high technology products like modern smartphones and self-driving cars, require that someone is able to represent the object. From how an object looks, to specifying the various materials and elements that comprise the object and how they are connected, to how the circuit boards are laid out for any piece of equipment that requires power to operate, the tools that enable a person to represent the physical system are at the core of digital engineering practices. This suite of tools includes both the software that helps a designer specify their design in order to communicate the design to others, run various analyses on their designs, as well as manage the the lifecycle of the design as it matures and evolves over time.

THE KEY AREAS OF THESE TOOLS ARE:

- **Computer Aided Design (CAD):** software used to create, modify, analyze and optimize a design
- **Electrical computer-aided design (ECAD):** software used to develop electronic systems including printed circuit boards (PCBs) and integrated circuits (ICs)
- **Product Lifecycle Management (PLM):** techniques and software which can integrate many aspects of a product's life cycle, including design, development, manufacturing, supply chain, distribution, marketing, and sales

Let's examine each area in more detail.

Physical System Representation: Computer Aided Design

Computer Aided Design (CAD) is where most engineering solutions for physical objects begin to take shape. Often used to make a 2D sketch come to life in a 3D model, and sometimes used for the sketching from the very start, CAD offers an extremely robust set of tools to help the engineer conceive, develop, and share complex physical forms. CAD as we know it today, started at MIT in 1960 with a program called Sketchpad that created 2D representations and used a lightpen for user interaction. For those of you born after the introduction of the mouse and who have never used a lightpen, they were the original form of direct-manipulation technology that worked in a similar way to the interaction of our fingers on today's smartphones.

CAD has evolved significantly since the 60s and is now a mature and incredibly powerful technology. Far from being just a digital drafting board, CAD has become a powerhouse for creating, analyzing, and optimizing designs in a virtual space. At its core, CAD allows engineers to construct detailed 2D and 3D representations, manage design data, and explore countless "what if" scenarios—all before a single prototype is built. It's not just a single tool; it's the framework on which much of modern design and innovation rests.

A key feature in modern CAD systems is parametric modeling. Traditionally if you used CAD to generate the form of, say, a guitar neck, you might define the specific dimensions of the length, width, thickness and cross-section of a guitar neck. If you changed one dimension, the rest would stay static of course and the guitar neck might look very odd. However with parametric modeling, you define the relationships between the dimensions. In this case, if you make a single change to a parameter (such as the length of the neck), the whole model would adapt.

Parametric modeling often, but not always, requires a little more time up front when developing the model. But the payoff is enormous both in being a huge time saver later down the line but also for maintaining integrity in large, interconnected designs while helping engineers experiment and iterate more rapidly. This approach is critical in highly competitive industries like automotive manufacturing. Think about a car company that develops a platform on which they construct many different cars: the short design is for the compact, lengthen some parts of it a bit and you have mid size, make it a little larger and a little wider, and you have an SUV. By creating a platform design in which some things change together in pre-defined ways, while other elements stay static, allows the engineer to rapidly experiment with a variety of designs to develop better and more interesting cars that all leverage a single platform. It also helps engineers conceive of new platforms that can allow for the creation of cars that might be gas powered and need to carry an engine or are purely electric and need to carry batteries.

What drives the engineer of the future is the ability to understand the deep technical issues while simultaneously considering and codifying the business and user drivers. When an engineer goes beyond simply solving the technical problem but also considers things like manufacturability, industrial design, shipping and installation from the beginning, they will produce designs that not only address the current objectives but the future objectives as well. In our music analogy, it would be like a composer writing for a violin where they want the player to place the bow on 2 strings that aren't touching each other while not playing the string between them. The design of the score might create a beautiful sound but it may not be playable by a single musician. For our engineer they might consider things like: can a person build it with their hands? Can a robot build it with a 6-axis robot arm?

The future engineer will not only be required to know how to articulate their ideas in ways the more resemble code, become proficient in collaborative digital platforms that unify CAD with aspects of Product Lifecycle Management (PLM) and Simulation technology, and consider the end of the product life cycle from the start of the project.

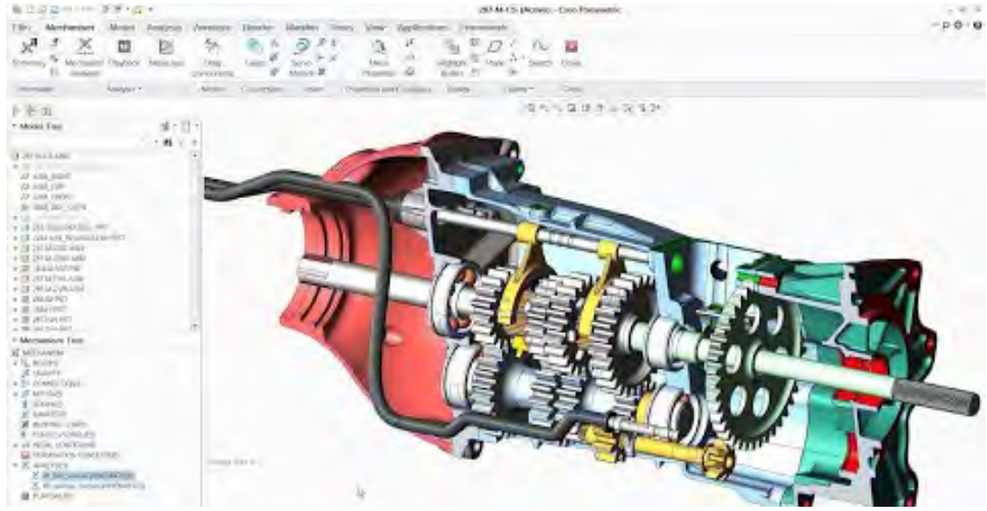
The engineer of the future will need to connect the output of their CAD systems to other systems, and here are some of the essential data they will need to extract from CAD models to create robust digital threads:

- Attributes of the model which include data such as the density, volume, surface area, and mass of objects.
- User defined parameters that allow parametric modifications inside the model which could data things such as the length of the wing, diameter of rotor, bolt hole pattern dimensions.
- A set of views including: isometric, top, bottom, left, right, front, and back views as well as a 3D file such as an .obj.
- A Bill of Materials (BOM)

Here are some examples of popular CAD systems, and a few notes about each one as well as screenshots.

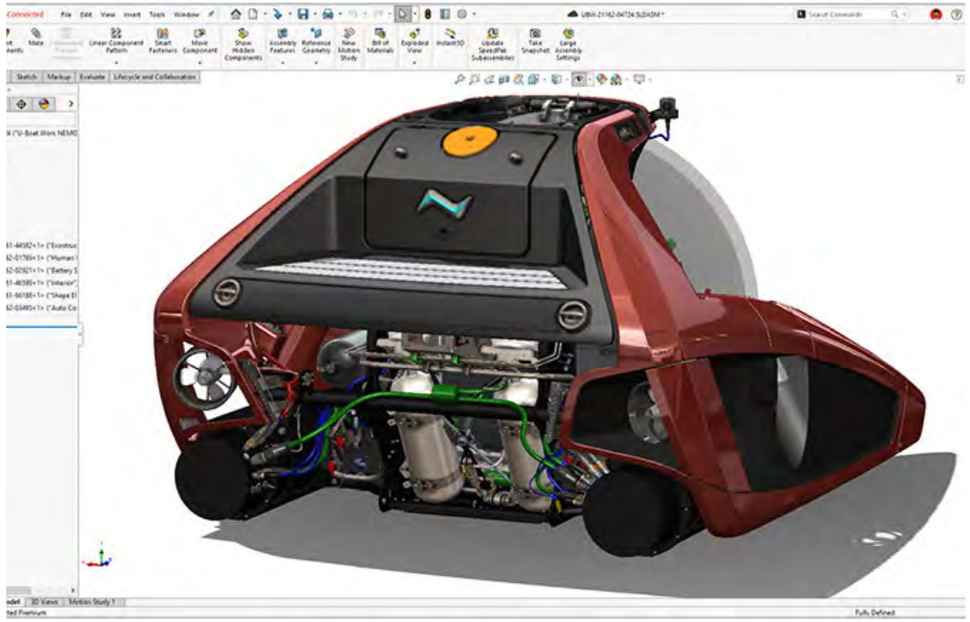
PTC CREO PARAMETRIC

PTC Creo Parametric (CREO), formerly known as Pro/Engineer, is a fully parametric modeling package developed by PTC that supports large assemblies and is used by leading companies around the world. It combines solid, surfacing, and polygonal modeling approaches with an advanced generative design and surfacing modules. It includes add-on packages like advanced surfacing, a 3D printing slicer, GD&T, mold design, and simulations.



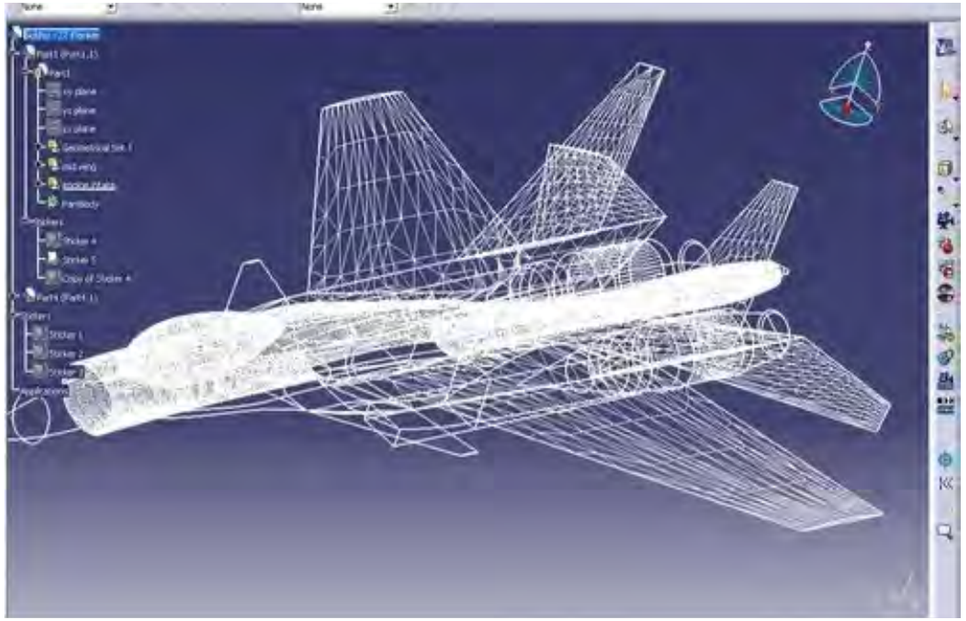
DASSAULT SOLIDWORKS

Dassault SolidWorks is one of the most widely used professional 3D CAD tools globally. It provides powerful solid and surfacing tools, including lofts, sweeps, and boundary surfaces, as well as modules for sheet metal, simulation, mold flow analysis, and rendering. It remains popular due to these robust features and accessibility, especially with affordable options for students.



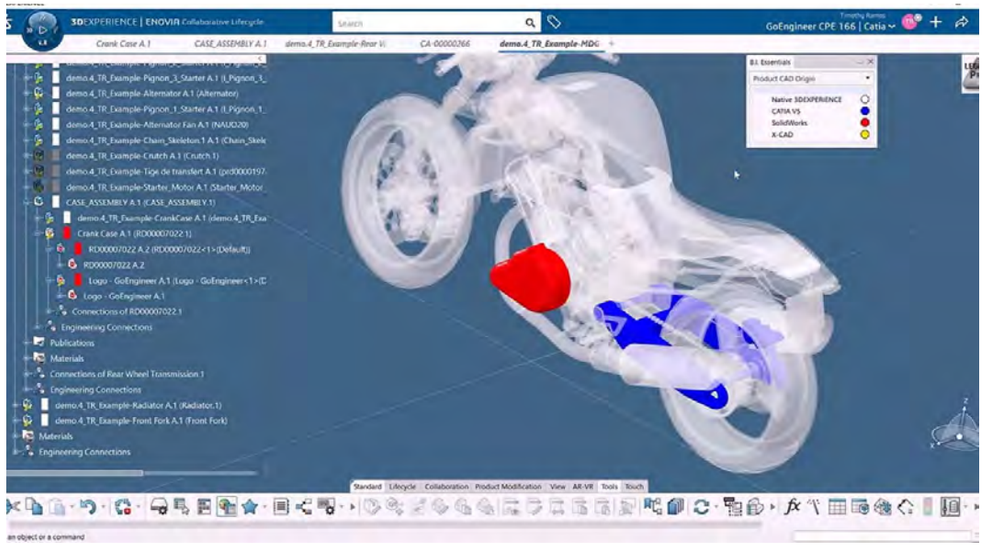
DASSAULT CATIA V5

Dassault Catia v5 developed by Dassault Systèmes, serving as a tried and true multi-platform CAD/CAM/CAE software suite for more than two decades. CATIA v5 is utilized for 3D product design and engineering, particularly for industries like aerospace and automotive and for complex product development.



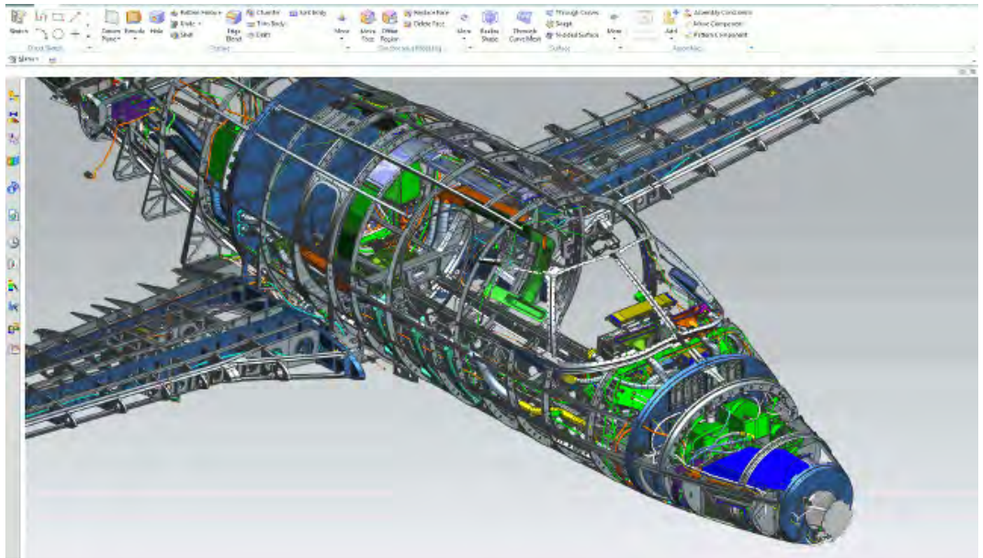
DASSAULT 3DEXPERIENCE CATIA

Dassault 3DExperience Catia is the latest iteration of CATIA (formerly called v6) which is integrated into the 3DEXPERIENCE platform by Dassault Systèmes. It combines design, simulation, and collaboration tools on a cloud-based platform to enhance cross-discipline collaboration and data management; it's also popular in the aerospace and automotive industries for complex product development.



SIEMENS NX

Siemens NX, formerly known as Unigraphics, is a versatile CAD software providing advanced part design, assembly, and drafting features; like its competitors, it can be paired along with high-level PLM, CAM, and CAE solutions to enhance its capabilities. It is widely adopted by leading companies, and features hybrid modeling with freeform, solid, and first class surfacing methods. NX includes Synchronous Technology for intelligent direct editing and offers an adaptive user interface that predicts frequently used commands, making it particularly user-friendly. Overall, NX remains a powerful and innovative tool in the CAD industry.



Physical System Representation: Electronic Computer-Aided Design (ECAD)

Electronic Computer-Aided Design (ECAD) is software used for designing and developing electronic systems—from simple printed circuit boards (PCBs) to complex integrated circuits (ICs). Just as CAD revolutionized mechanical design, ECAD has transformed the electronics industry by enabling engineers to create intricate designs with precision and efficiency.

The origins of ECAD trace back to the 1960s, when the burgeoning field of electronics demanded more sophisticated design tools. Before ECAD, engineers had to manually draft circuit diagrams – a time-intensive and error-prone process. It involved meticulous hand-drawing of circuits and components on large paper sheets. The advent of transistors and integrated circuits highlighted the need for a more efficient way to design circuits.

Over the decades, ECAD tools have evolved significantly, incorporating advanced features like simulation, 3D modeling, and collaborative capabilities. Today's ECAD tools are highly sophisticated, integrating multiple functionalities into a single platform. They offer schematic capture, PCB layout, component placement, routing, and even thermal and signal integrity analysis. Advanced ECAD systems enable 3D visualization of electronic assemblies, bridging the gap between electronic and mechanical design—a crucial aspect as devices become more compact and integrated.

One transformative capability of modern ECAD tools is the integration of simulation and analysis directly into the design environment. Engineers can perform circuit simulations to validate functionality, analyze signal integrity to prevent issues like crosstalk or electromagnetic interference, and run thermal simulations to ensure proper heat dissipation. This holistic approach reduces the need for multiple prototypes, saving time and resources.

In the evolving landscape of digital engineering, the engineer of the future will need to be conversant with the blurry line between mechanical engineering and electrical engineering, understanding electronic design techniques and strategies as well as how those systems interact with pure mechanical systems. As devices become more interconnected and complex the need for the future engineer to be able to connect these disciplines will be crucial.

For many years before the advent of jazz, the instruments that make up a jazz trio existed and were well understood (a piano, double bass and drums). As the genre of jazz evolved the interplay between these instruments became well understood, leading to the creation of jazz greats such as the Nat King Cole Trio in the late 1930s, the Bill Evans Trio in the late 1950s, to the currently performing Brad Mehldau Trio.

Much like our engineer of the future, we could never imagine a famous jazz music producer who only knew about getting a great piano performance – but had no awareness about how to also get a great drum, and double bass performance. Understanding the interplay between disciplines that are closely related – in this case between the macro world of mechanical engineering and the micro and quantum world of electronics engineering, enables the engineer of the future to go beyond what the very best companies in the world are capable of doing today.

Advanced technologies such as artificial intelligence and machine learning are beginning to make their way into ECAD tools, offering features like automated routing, design optimization, and error detection. Cloud-based platforms facilitate collaboration across global teams, breaking down barriers of geography and time zones.

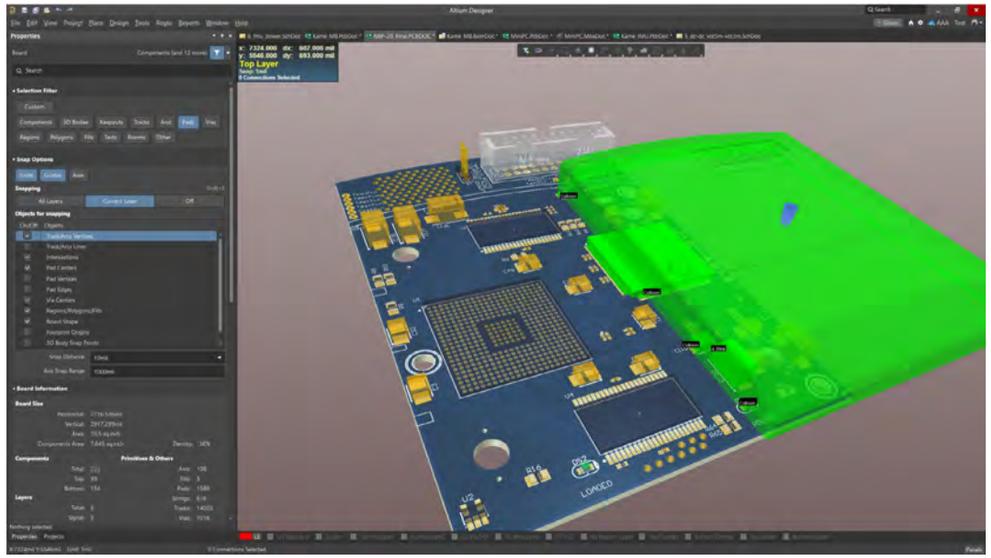
To create robust digital threads that connect ECAD systems with other engineering tools, engineers will need to extract and manage a variety of data:

- **Schematic Designs:** Detailed circuit diagrams showing electrical connections and components.
- **Bill of Materials (BOM):** Comprehensive lists of all components required, including part numbers, specifications, and sourcing information.
- **PCB Layout Files:** Data defining the physical layout of the circuit board, including layers, traces, vias, and component placement.
- **Netlists:** Lists of electrical connections used for simulation, testing, and verification.
- **3D Models:** Visual representations of electronic components and assemblies for integration with mechanical designs.
- **Simulation Data:** Results from signal integrity, power integrity, thermal, and electromagnetic simulations.
- **Manufacturing Files:** Such as Gerber files, drill files, and assembly drawings required for fabrication and assembly.
- **Design Rules and Constraints:** Specifications governing the physical and electrical parameters of the design.
- **Revision History:** Version control information tracking changes and updates to the design.

Here are some examples of widely used ECAD systems, each offering unique features and capabilities:

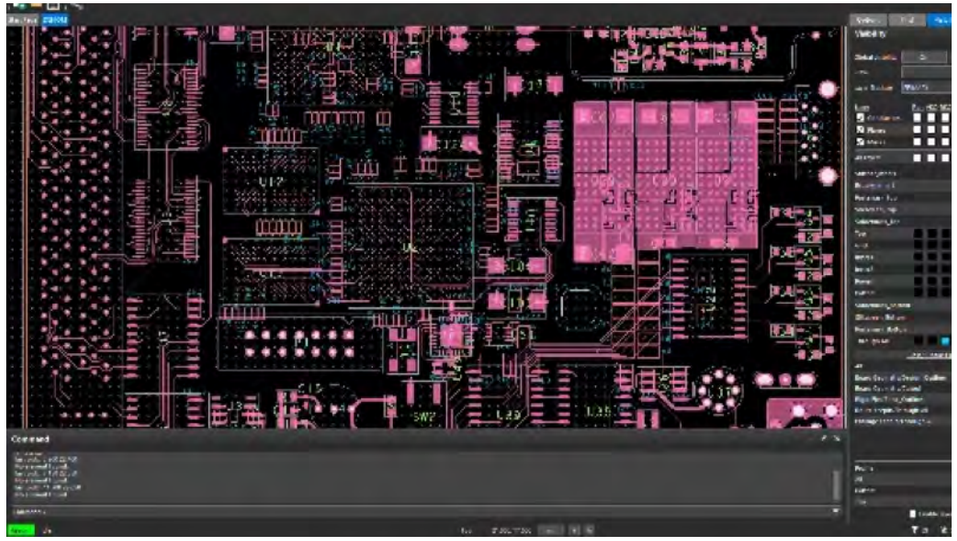
ALTIUM DESIGNER

Altium Designer is a comprehensive ECAD solution that integrates schematic capture, PCB layout, and FPGA development within a unified design environment. It offers advanced 3D visualization and clearance checking, making it easier to detect mechanical conflicts early in the design process. Altium Designer supports collaboration through its Altium 365 platform, facilitating data management and version control.



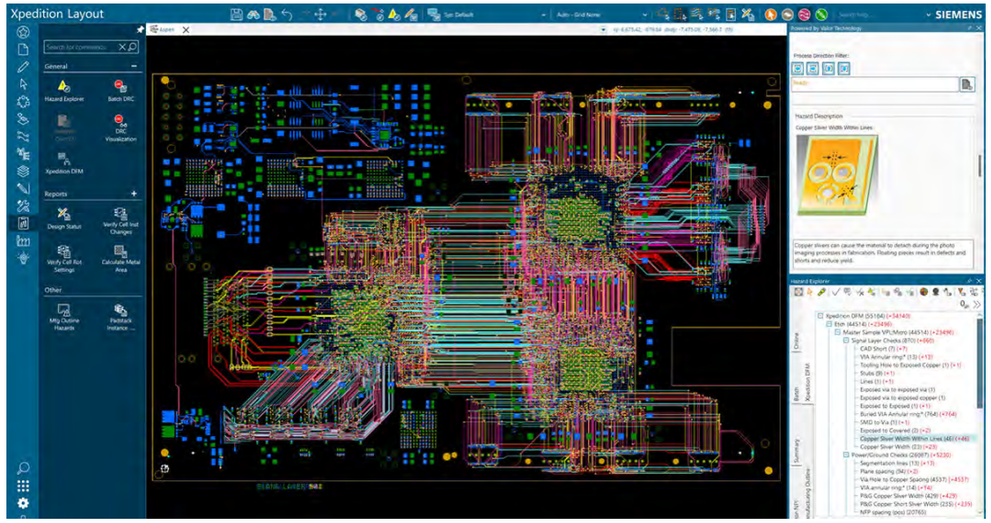
CADENCE ORCAD AND ALLEGRO

Cadence offers a suite of ECAD tools with OrCAD focusing on smaller designs and Allegro catering to high-end, complex projects. OrCAD provides robust schematic capture and PCB layout capabilities suitable for a wide range of applications. Allegro extends these capabilities with advanced features for high-speed and RF designs, including signal and power integrity analysis. Both tools are integrated within the Cadence platform, supporting simulation and verification workflows.



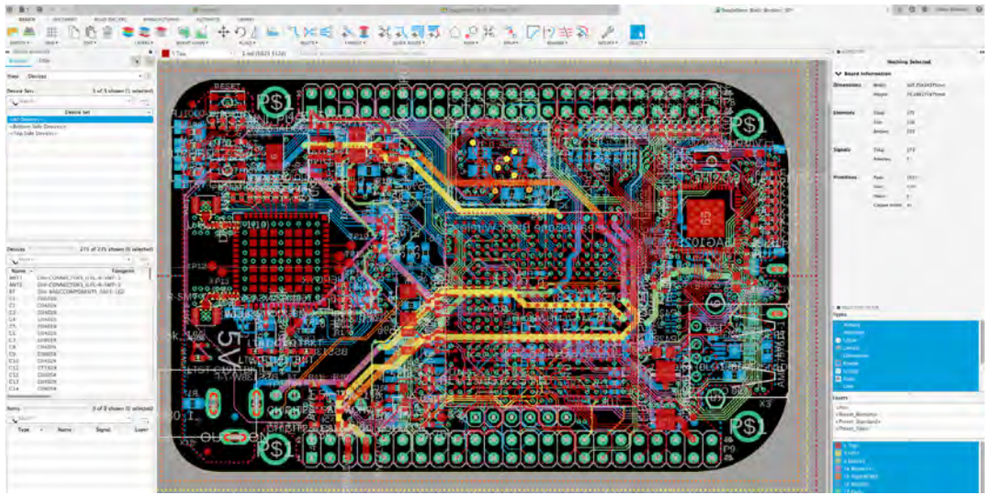
SIEMENS EDA (MENTOR GRAPHICS) PADS AND XPEDITION

Siemens EDA, formerly Mentor Graphics, offers PADS for small to mid-size projects and Xpedition for enterprise-level designs. PADS is known for its user-friendly interface and strong community support, making it accessible for both beginners and professionals. Xpedition provides advanced features such as multi-board systems design, 3D layout, and comprehensive validation tools.



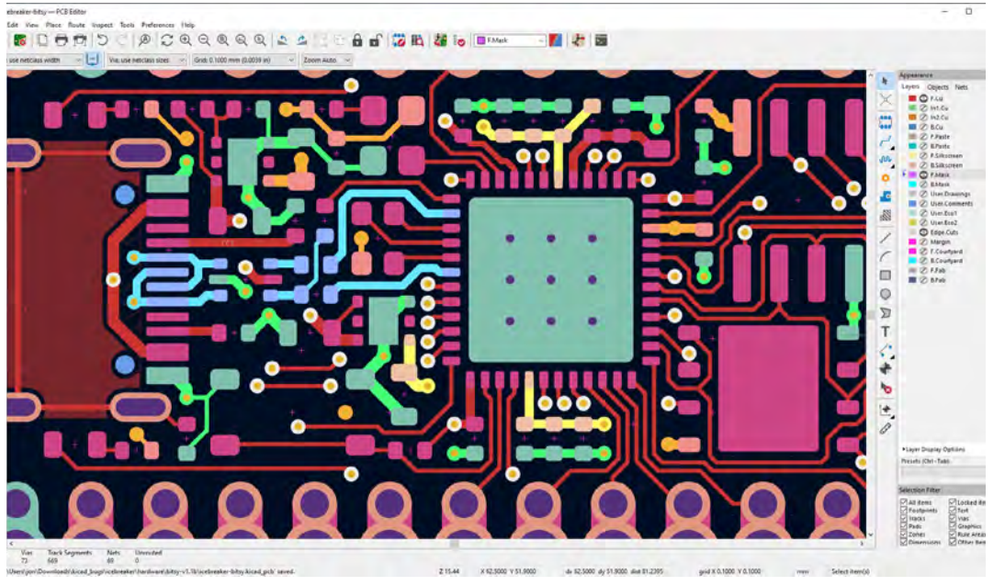
AUTODESK EAGLE

EAGLE (Easily Applicable Graphical Layout Editor) is an ECAD tool offering schematic capture and PCB layout functionalities. Now part of Autodesk, EAGLE integrates with Fusion 360, bridging the gap between electronic and mechanical design. This integration allows better collaboration between ECAD and MCAD, facilitating the design of complex electromechanical systems.



KICAD

KiCad is an open-source ECAD suite that provides professional-grade features without licensing costs. It includes schematic capture, PCB layout, and a 3D viewer for PCB assemblies. KiCad has gained popularity due to its active community and continuous development, making it a viable option for both hobbyists and professionals.



Physical System Representation: Product Lifecycle Management (PLM)

Product Lifecycle Management (PLM) is a framework where the entire lifecycle of a product—from initial concept to end-of-life—can be managed and optimized. PLM encompasses a suite of software solutions and collaborative processes that integrate people, data, processes, and business systems. The roots of modern PLM actually start in the late 70s as an outgrowth of CAD systems, and by the 1990 the systems were mature enough to enable Boeing to create the Boeing 777, the first aircraft entirely designed using 3D CAD and PLM tools.

Modern PLM systems are the connective tissue that binds elements of the design and manufacturing ecosystems, providing a single repository where data generated from many other tools are stored. Engineers and non-engineer stakeholders from different departments within an organization can access this central data store in order to collaborate.

One of the most powerful features of PLM is version control. Before PLM systems, people would manage the data with spreadsheets which of course introduced lots of errors. Modern PLM systems ensure that design iterations and revisions are clear to viewers and easy to manage consistently. So when a change is made to a particular component, a PLM system can cascade updates throughout related documentation which can help keep stakeholders aligned. However there is a limitation of PLM systems in complex engineering projects: they don't connect across organizational boundaries. When that happens this is where we're back to spreadsheets and where we eliminate the ability for engineers to collaborate effectively.

Bringing this back to our music analogy, consider a band who makes a recording in a studio where all the tracks are brought together to form a single released recording. In November of 1966 the Beatles recorded Strawberry Fields Forever and developed a huge number of recordings, with many variations. If we imagine John Lennon as our engineer working with the PLM, he would create and label many versions of his recordings. The same would be true for the other "engineers" (Paul, George and Ringo). But what brought all these versions together to become something more than the sum of their parts? Their extremely talented producer, George Martin, could work across the divide between the musicians, and because he understood the objectives of each of them he could effectively connect their work. By the time they got to take 26, George Martin added 3 cellos and 4 trumpets creating part of the masterful work we know today.

The engineer of the future will have to work like George Martin and go beyond the current PLM framework. This is where the concept of digital threading becomes critical to working more efficiently across organizational and geographic boundaries.

The strength of a PLM system is that it can maintain all the version information about any model or document that's been added to the system, helping users understand if they are looking at a formally approved baseline, a preliminary draft, an approved version that's been accepted by stakeholders, a final version, or even a snapshot in time that can be arbitrarily saved for reference.

The engineer of the future needs to understand how various PLMs will need to connect with each other in order to gain the efficiencies of robust collaboration. By considering strategic business objectives, the engineer can deploy robust digital threads to enable mission critical objectives at scale, including: ensuring that proprietary techniques and processes are shielded while relevant data is freely shared, effectively coordinating suppliers and manufactures who may be in competitive markets, and maintaining a clear audit trail of changes that occur between organizations.

The engineer of the future will need to connect the output of their PLM systems to other systems by creating comprehensive digital threads that enhance collaboration and lifecycle management. Some valuable data to be extracted from PLM systems include:

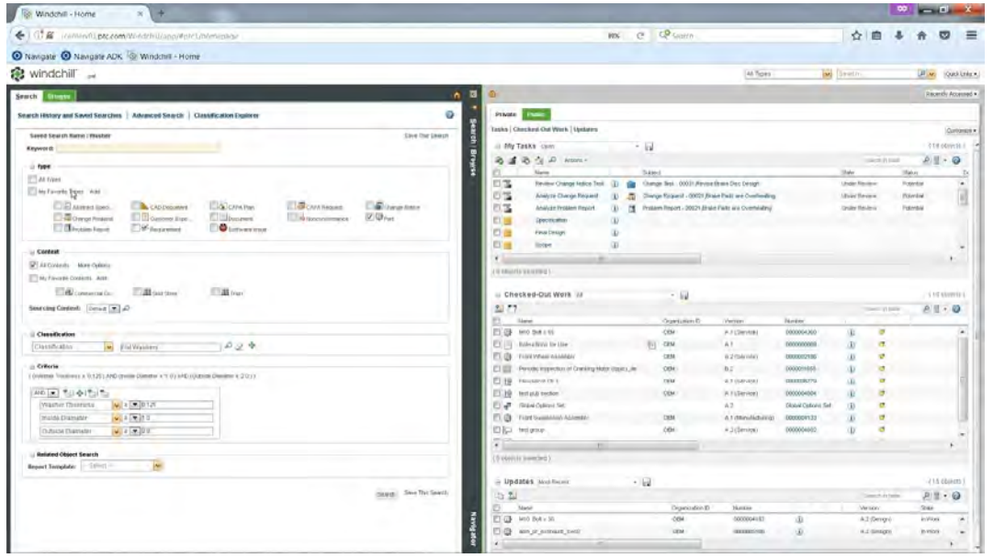
- **Product Metadata:** Including product ID, version history, and lifecycle stage (e.g., conceptual, prototype, production)
- **Requirements and Traceability Data:** documentation of customer and regulatory requirements, with links to CAD models and test results that verify these requirements are met
- **Change Management Records:** logs of design revisions and change orders, capturing the who, what, when, and why behind modifications to ensure accountability and version control

- **Bill of Materials (BOM):** this can include raw materials, subassemblies, and finished components, linked with supplier information and cost data
- **Supplier and Procurement Data:** information on part sourcing, supplier certifications, lead times, and procurement strategies
- **Simulation and Analysis Data:** results from simulations
- **Compliance and Certification Records:** documentation confirming adherence to industry standards and certifications, used for auditing and market approval
- **Manufacturing and Process Planning:** steps for manufacturing, including assembly instructions, process simulations, and quality checks to bridge the gap between design and production.
- **Service and Maintenance Schedules:** data that will inform downstream activities like preventive maintenance, warranty management, post-production updates and more
- **Digital Twin Data Integration**: links from digital models to data that come from physical prototypes and products that enable ongoing performance monitoring and future design improvements

Here are some examples of popular PLM systems, and a few notes about each one, as well as screenshots:

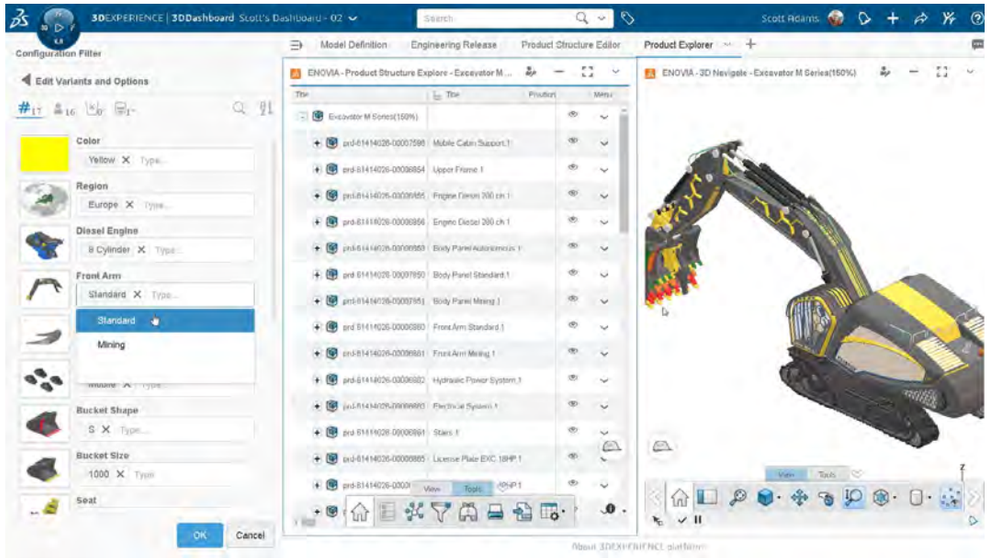
PTC WINDCHILL

PTC Windchill is strong in product data and change management, known for its comprehensive CAD integration and BOM management and product data management, particularly suited to manufacturing and engineering-heavy industries.. It also supports a scalable architecture that accommodates a variety of deployments, from on-premises to cloud-based, with modular options for organizations of different sizes and complexities. Its standout feature is IoT integration with PTC ThingWorx, making it ideal for connected product lifecycles.



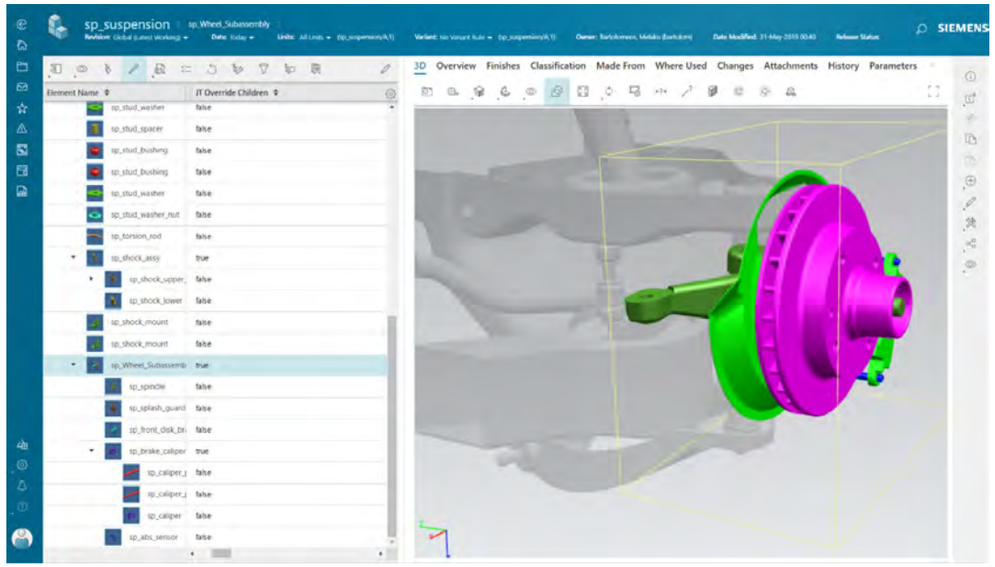
3DEXPERIENCE CLOUD PLM

3D Experience unifies PLM, CAD, and simulation on one platform with real-time, cloud-based collaboration. It is designed to integrate many different apps within the platform to provide a one stop shop for Product Design and team collaboration. It also boasts seamless integration with Dassault's CATIA and SOLIDWORKS tools.

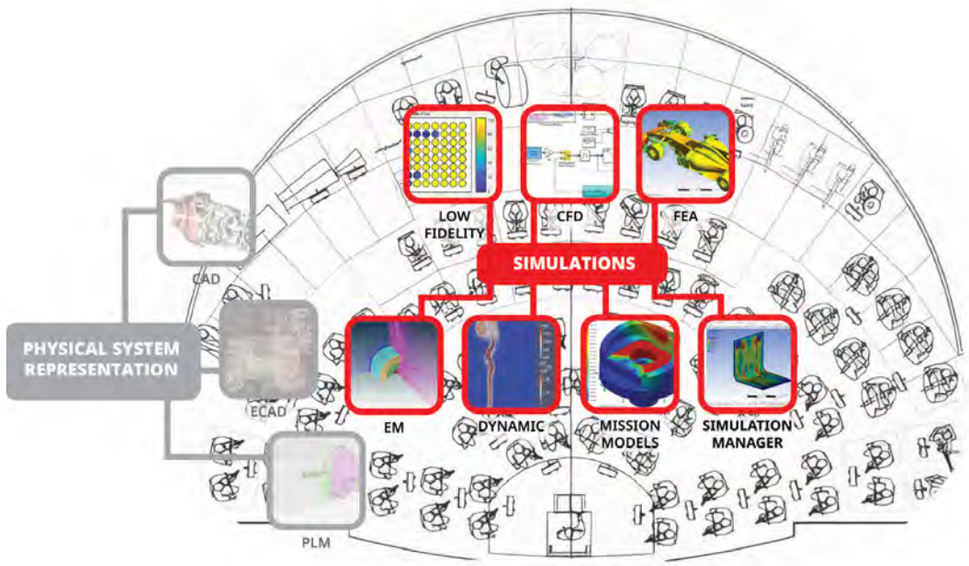


SIEMENS TEAMCENTER

Siemens Teamcenter is known for its enterprise-grade scalability and strong multi-CAD support, making it versatile for global operations. It excels in full lifecycle management, supporting complex product configurations and variant management across large teams.



Simulations



Simulation software attempts to replicate the behavior of real-world physical systems under various conditions in a virtual environment. They help engineers perform virtual testing of a product to, say, see how it might perform in high temperatures, under extreme loads, or while experiencing rapid movements. They help engineers understand how fluids move in order to simulate the airflow over an airplane wing or across a cooling device for a car or a piece of electrical equipment before the design is built. And they help engineers even understand how to modify the operation of something after it's built.

Simulation software has its origins in the 1960s in which Boeing and NASA simulate various stresses, vibration effects and heat effects on aircraft and spacecraft. By the 1970s, CAD software began incorporating simulation software in order to more tightly connect the dots between design and testing without having to produce a physical prototype early in the process.

Modern simulation software can simulate a vast array of physical and operational conditions, including structural stress and strain, heat transfer, fluid dynamics, electromagnetic fields, crash impacts, and acoustic behavior. They also enable engineers to explore motion analysis, material behavior, chemical reactions, and manufacturing processes.

The best tools can even perform multiphysics simulations in which a few different simulations need to be performed simultaneously. To create an optimal design of an electronic motor for a car, an engineer needs to perform structural analysis, a thermal analysis and an electromagnetic analysis – each of which will interact and inform the other.

When designing a satellite, it's critical to consider the environment: space. In space there's no protection from solar radiation, so a satellite experiences extreme thermal loads and significant temperature changes as it turns toward and away from the sun. When the simulation runs to determine the heating and cooling effects, it will inform the structural model. That model can then calculate the growth and shrinking of the various materials that have been specified, in order to show the stresses on the structure. We've made satellites for a long time, so it might appear that this knowledge should be codified without the need for lengthy simulation times, however, sometimes the slightest change can cause a huge and unintended downside effect. If you're making a sensitive optical system that's meant to look across the universe, a slight distortion on the part of the satellite that holds a mirror, on the order of a few nanometers (about 100,000 times thinner than a human hair), can destroy the entire value of the satellite. However, if you can accurately simulate the system before building it, you can design better solutions at a significant cost and time savings.

The interesting thing about simulations is that they are just that – a simulation. They can never account for all the factors in the real world, which is why prototypes are still produced. Performing tests on the physical model can show you where the simulation and the physical model are mis-aligned and help you rationalize the physical system with the virtual representation. This is how you build extreme confidence that when the system is finally built it will perform correctly and the model can even be used to predict how a change in the operation of the system (in this case, a satellite that is already launched into low-earth orbit) will affect how the system will work in real life.

To further extend our music analogy, a simulation is like using that K-250 synthesizer we talked about before. It can sound a lot like an orchestra, however, there's a huge difference

between Hans Zimmer simulating the score of the Lion King or Interstellar on a set of synthesizers, and the sound we hear from a live orchestral recording. In fact for any set of musicians playing together, there's a cohesiveness that only can occur when they've played together over a long period of time. And like simulations, more iterations and more connections between the virtual and physical representations help to refine the effectiveness of the simulation.

The engineer of the future will need to understand how multiphysics simulations work and be able to create a digital thread of a hierarchical data pipeline that represents how each simulation should connect and inform another simulation, in order to produce a high quality digital representation of a new design.

The engineer of the future will need to know how to properly thread simulations to various data sources in order to build Monte Carlo simulations that run thousands of scenarios with random changes. By doing this effectively, this engineer of the future can identify worst-case scenarios and use an anti-optimization approach to minimize the risk of catastrophic errors. This ability to integrate and analyze diverse data will reduce uncertainty, enhance the safety and reliability of complex systems, and minimize catastrophic errors

There are a variety of simulation tools, from low fidelity to high fidelity tools. The engineer of the future will need to connect the output of their simulation tools to other systems by creating comprehensive digital threads that enhance collaboration and provide human-in-the-loop iteration. Some valuable data to be extracted from simulation tools include:

- **Simulation Results and Performance Data:** Detailed outputs showing how a system or component behaves under various conditions, crucial for verifying design feasibility and performance
- **Input Parameters and Assumptions:** Documentation of the initial conditions, boundary conditions, and assumptions used in simulations for reproducibility and traceability
- **Failure Analysis and Stress Testing Data:** Information on stress points and potential failure modes, aiding in risk assessment and design improvements
- **Optimization Outcomes:** Data highlighting optimized solutions from parametric studies, which help guide design iterations
- **Model Interactions and Dependencies:** Details that connect simulation models with CAD, MBSE, or PLM systems, ensuring cohesive workflows and data integration across platforms

There is a diverse set of sub-disciplines of simulation tools that cater to different aspects of system performance, addressing needs from early conceptual validation to mission-critical analyses. Here are key sub-disciplines of simulation tools in digital engineering:

- **Low Fidelity Simulations** which strike a balance between speed and accuracy, making them ideal for preliminary design iterations
- **Computational Fluid Dynamics (CFD)** tools enable engineers to analyze fluid flow, heat transfer, and related phenomena
- **Finite Element Analysis (FEA)** is the cornerstone of structural analysis in engineering to model how a physical object will respond to real-world stresses
- **Electromagnetic (EM)** simulation tools show how electric and magnetic fields behave in electronic systems
- **Dynamic Simulations** simulate the behavior of systems that change over time
- Mission Models are specialized simulations used to replicate complex, multi-system interactions
- **Simulation Managers** are used to connect a set of simulations and provide a unified interface to efficiently track results

These sub-disciplines form an interconnected suite of tools that underpin modern engineering practices, allowing designers and engineers to predict, validate, and perfect their work across a broad spectrum of real-world conditions. Let's examine each area:

Simulations: Low Fidelity Simulations

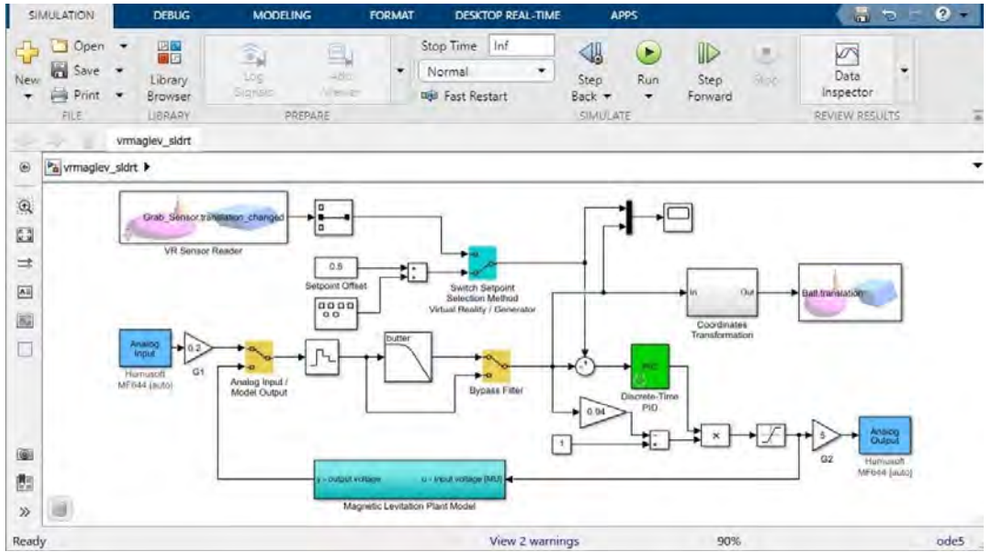
Low Fidelity Simulations serve as a first-pass tool for assessing designs without delving into complex details. They offer a quick way to validate concepts and identify glaring issues or feasibility concerns. Low fidelity tools are particularly useful in early-stage design, where broad strokes and rough estimates provide valuable insights before investing time in high-detail modeling. They strike a balance between speed and accuracy, making them ideal for preliminary design iterations. Here are examples of popular Low Fidelity Simulation tools:

MATHWORKS MATLAB

MathWorks MATLAB is a versatile programming and numeric computing environment widely used by engineers for data analysis, algorithm development, and numerical computation. It offers built-in functions for matrix manipulation, statistical analysis, and visualization, allowing engineers in fields such as electrical, mechanical, and civil engineering to model and solve complex mathematical problems. MATLAB's extensive libraries support tasks from signal processing and control design to image processing and optimization, making it valuable for prototyping, research, and creating custom analysis tools across engineering disciplines.

MATHWORKS SIMULINK

MathWorks Simulink, an extension of MATLAB, provides a graphical environment for modeling, simulating, and analyzing dynamic systems. Engineers use Simulink to build block-diagram models of systems that evolve over time, such as control systems, signal processing systems, and multi-domain physical systems. It enables engineers across fields like aerospace, automotive, and robotics to design, test, and optimize system behavior in a virtual setting, supporting rapid prototyping and model-based design. Simulink's simulation capabilities allow users to evaluate system performance under various scenarios, helping reduce development time and ensuring system reliability before physical testing.

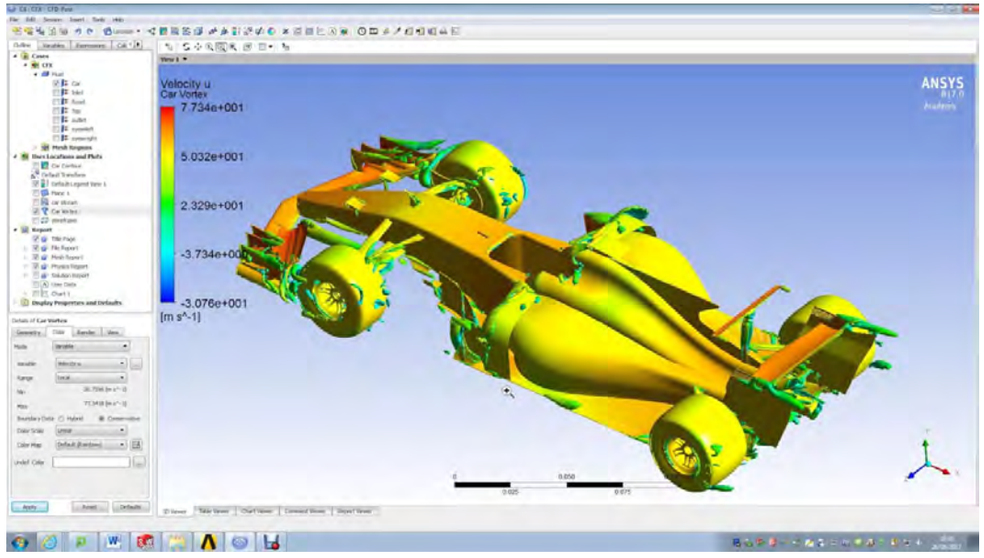


Simulations: Computational Fluid Dynamics (CFD)

Computational Fluid Dynamics (CFD) tools enable engineers to analyze fluid flow, heat transfer, and related phenomena within and around their designs. From predicting how air moves over an airplane wing to assessing cooling systems for electronics, CFD simulates real-world conditions to optimize designs for aerodynamics, efficiency, and thermal management. These tools employ complex numerical methods to solve the Navier-Stokes equations, offering high-fidelity insights into fluid interactions that would be challenging to observe through physical testing alone. Here are examples of popular Computational Fluid Dynamics Simulation tools:

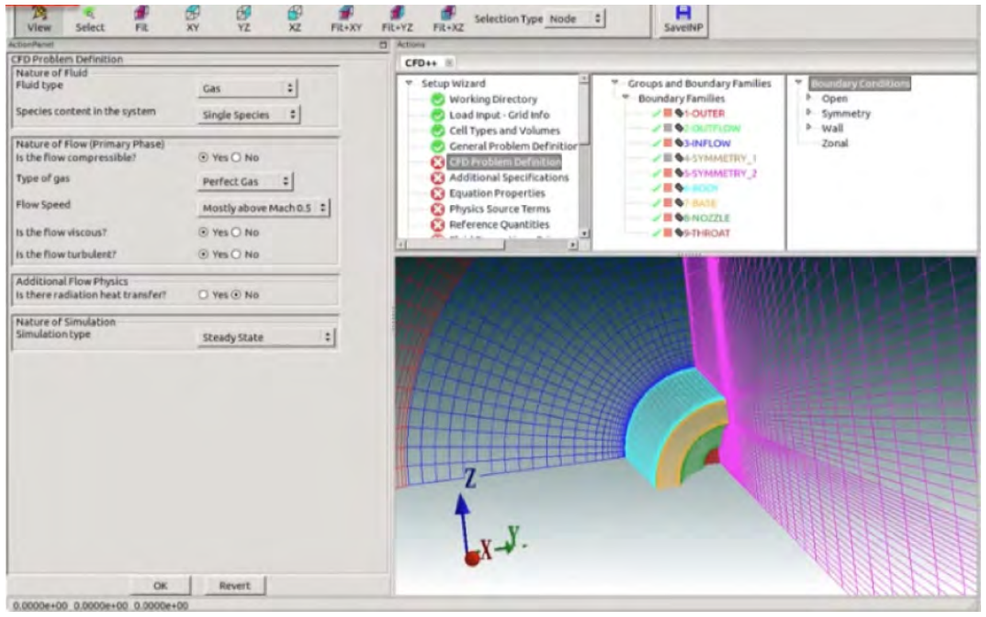
ANSYS FLUENT

Ansys Fluent is a computational fluid dynamics (CFD) software that enables engineers to model and simulate fluid flow, heat transfer, and chemical reactions. Commonly used in aerospace, automotive, and energy industries, Fluent helps engineers optimize designs for fluid dynamics and thermal management, improving performance and efficiency of systems such as engines, HVAC systems, and turbines.



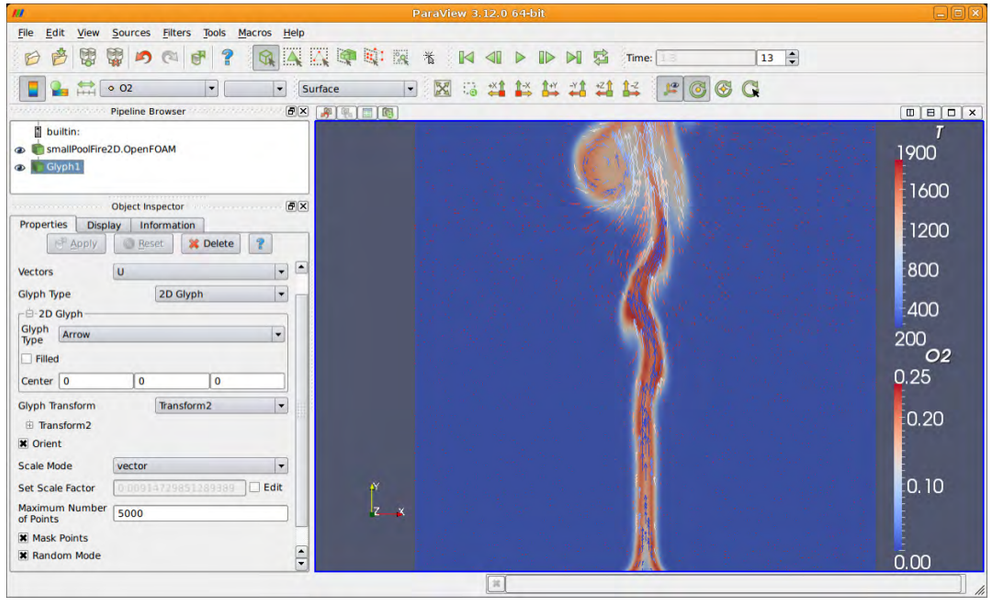
METACOMP ICFD++

Metacomp iCFD++ is an advanced computational fluid dynamics (CFD) software designed for simulating complex flow physics, including compressible and incompressible flows, multi-phase flows, and reactive flows. Known for its robustness and versatility, iCFD++ is widely used in industries such as aerospace, automotive, and energy, where engineers rely on it for high-fidelity modeling of aerodynamics, combustion, and thermal management. Its adaptive meshing and efficient solver capabilities make it suitable for challenging applications involving high-speed or turbulent flows, enabling detailed insights into fluid behavior in intricate scenarios.



OPENFOAM

OpenFOAM is an open-source computational fluid dynamics (CFD) software suite renowned for its versatility in simulating complex flow phenomena, including compressible and incompressible flows, multi-phase interactions, and heat transfer. Widely used in academia and industry, OpenFOAM supports a broad range of applications from aerodynamics and hydrodynamics to chemical processes and environmental modeling. Its modular structure allows users to customize solvers and boundary conditions to fit specific project needs, making it adaptable for high-fidelity analysis in fields such as aerospace, automotive, and energy. With features like automatic meshing tools and robust solvers, OpenFOAM provides a powerful platform for exploring intricate fluid dynamics and advancing engineering solutions.

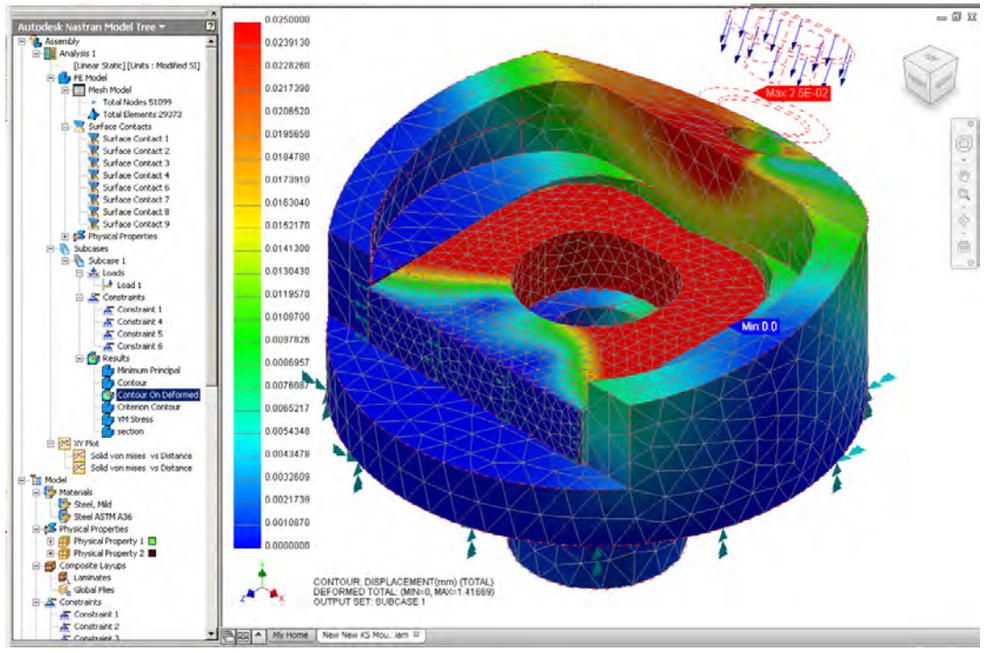


Simulations: Finite Element Analysis (FEA)

Finite Element Analysis (FEA) is the process of dividing a structure into smaller, manageable pieces called finite elements, which allow engineers to predict how complex materials and geometries will respond to physical forces such as tension, compression, and shear. This tool is invaluable for ensuring the safety and durability of products ranging from bridges and buildings to automotive parts and medical devices. FEA simulations help in identifying weak points and optimizing material use, reducing weight and cost while maintaining performance. Here are examples of popular Finite Element Analysis Simulation tools:

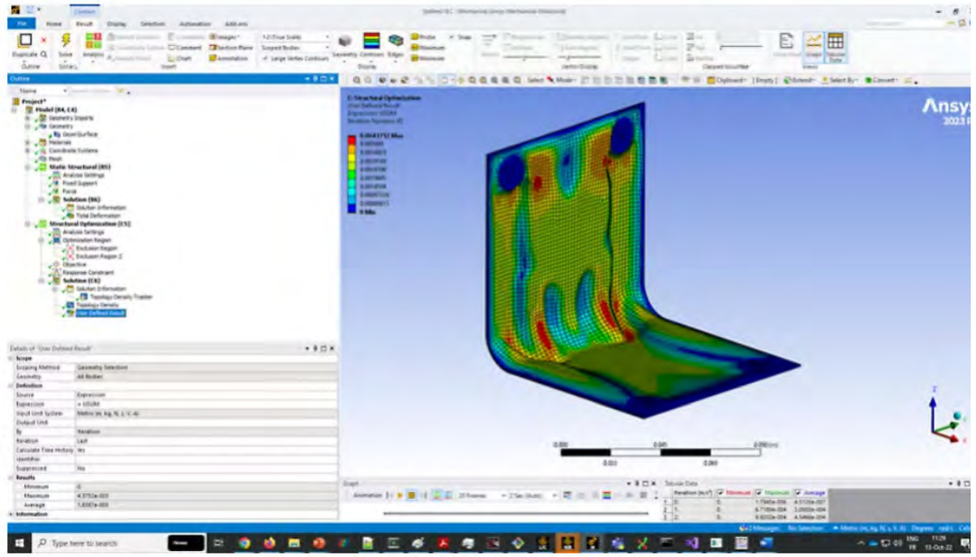
MSC NASTRAN

MSC NASTRAN is a finite element analysis (FEA) tool used by engineers to evaluate the structural performance of components and systems under various conditions. It supports a range of analyses, including linear and nonlinear static, dynamic, and thermal assessments, making it widely applicable across industries like aerospace, automotive, and civil engineering for optimizing designs and ensuring structural integrity.



ANSYS MECHANICAL

Ansys Mechanical is a comprehensive FEA tool that allows engineers to simulate static and dynamic mechanical phenomena, such as stress, deformation, and thermal effects on parts or assemblies. It is frequently used in product design and optimization across sectors like automotive, aerospace, and industrial manufacturing to validate and improve the durability and efficiency of mechanical components.

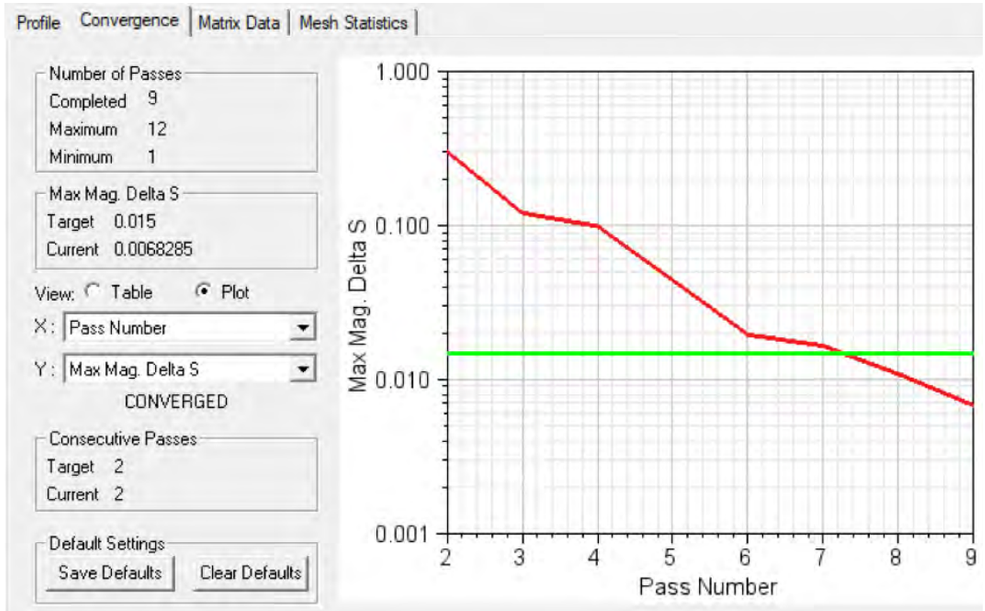


Simulations: Electromagnetic (EM) Simulation Tools

Electromagnetic (EM) simulation tools enable the engineer to understand the behavior of electric and magnetic fields within and around electronic systems. These tools are used to ensure that antennas, wireless communication systems, and PCB layouts perform as expected. High-frequency signal analysis, electromagnetic compatibility (EMC), and interference prediction are all addressed with EM simulation. As electronics become more compact and powerful, the need for precise EM analysis to mitigate signal degradation and crosstalk is increasingly essential. Here are examples of popular Electromagnetic simulation tools:

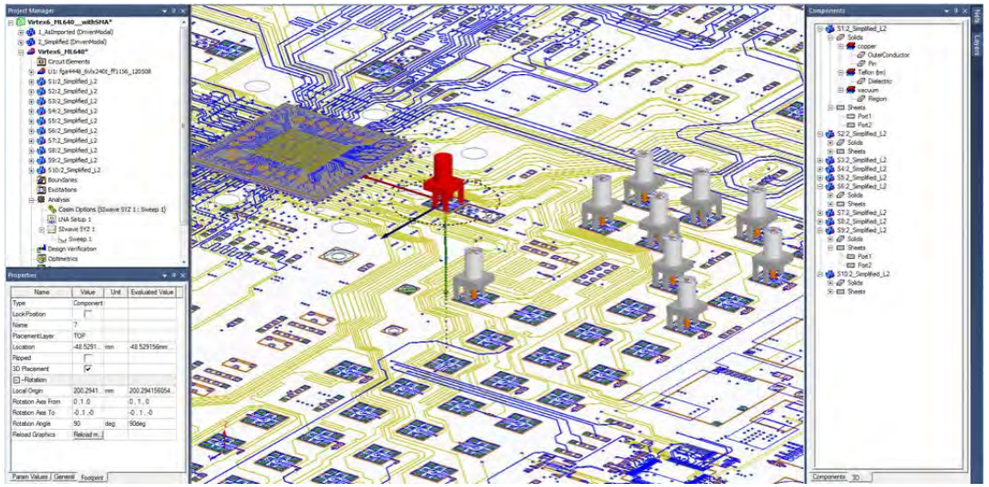
ANSYS HFSS (HIGH-FREQUENCY STRUCTURE SIMULATOR)

Ansys HFSS (High-Frequency Structure Simulator) is an advanced electromagnetic simulation tool developed by Ansys, used for designing and analyzing high-frequency components such as antennas, RF/microwave devices, and complex electronic structures. HFSS employs a 3D full-wave electromagnetic field solver, enabling accurate modeling of electromagnetic behavior and signal integrity. It supports the simulation of complex structures, including multi-layer PCBs, packages, and integrated circuits, with precise results that are critical for the development of high-performance systems. HFSS integrates with other Ansys tools to support multi-physics simulations and allows for workflow collaboration, although real-time collaborative editing features may be limited compared to tools designed for concurrent team-based development. This makes HFSS particularly valuable in industries such as telecommunications, aerospace, and automotive, where electromagnetic performance is critical.



ANSYS ELECTRONIC DESKTOP

Ansys Electronic Desktop provides a suite of tools for electromagnetic, circuit, and system simulations, allowing engineers to model and analyze electronic and RF (radio frequency) devices. It's used by electrical and electronics engineers in industries like telecommunications, automotive, and consumer electronics to ensure reliable performance of antennas, PCBs, and high-frequency devices.



Simulations: Dynamic Simulations

Dynamic Simulations are used to simulate the behavior of how a system changes over time, capturing the interactions between various components in motion. They provide insights into how mechanical systems respond to forces, torques, and movements. For example, dynamic simulations can assess vibrations in an engine, the response of suspension systems in vehicles, or the motion of robotic arms. By modeling these time-dependent behaviors, engineers can refine designs for better stability, performance, and reliability. Here are examples of popular Dynamic Simulation tools:

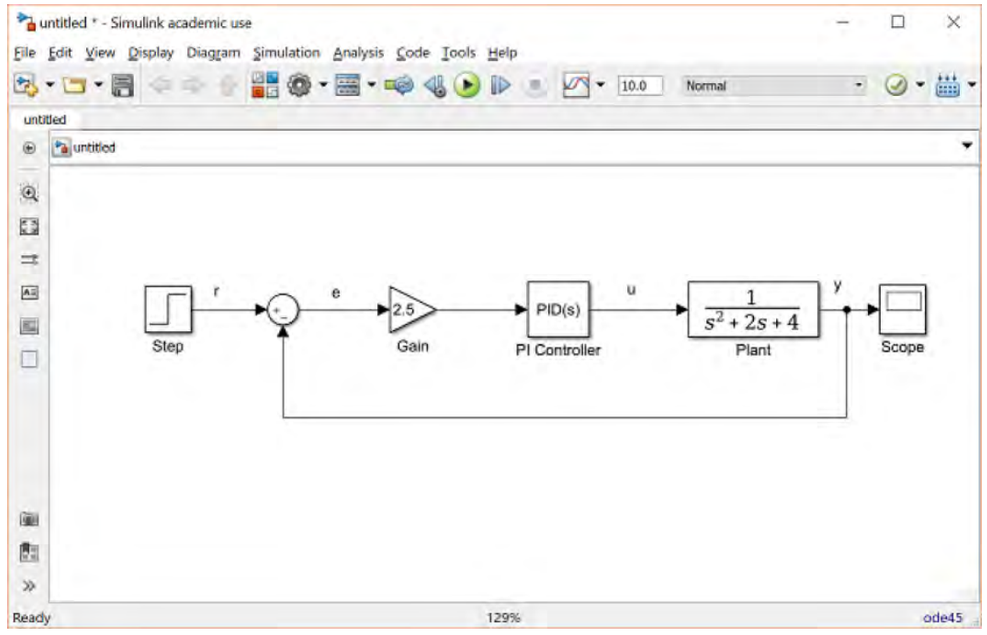
C++ (INCLUDING OPEN SOURCE LIBRARIES)

Eigen is an open-source C++ template library known for its efficient handling of linear algebra, including matrices and vectors, making it essential for robotics and vehicle dynamics calculations where real-time performance is critical. ODE (Open Dynamics Engine) is another widely used open-source library focused on simulating rigid body dynamics, providing realistic motion and interaction modeling in robotics, vehicle simulation, and complex mechanical systems.

```
#include <iostream>

using namespace std;

// One function works for all data types.

template <typename T>
T tMax(T x, T y)
{ return (x > y)? x: y; }

int main()
{

cout << tMax<int>(3, 7) << endl; // Call myMax for type int
cout << tMax<double>(3.0, 7.0) << endl; // call myMax for type d
cout << tMax<char>('g', 'e') << endl; // call myMax for type char

return 0;
}
```


MATHWORKS SIMULINK

It is worth revisiting **MathWorks Simulink** in the context of Dynamic Simulations. It provides an intuitive block diagram interface that allows engineers to model, simulate, and analyze systems that exhibit dynamic behavior, such as control systems, vehicle dynamics, and robotics. With built-in support for 3DOF and 6DOF simulations, Simulink can accurately represent the linear and rotational movements of complex systems, enabling comprehensive studies of their performance under various conditions. Its integration with toolboxes like the Aerospace Blockset enhances its capabilities for flight and vehicle dynamics, while real-time simulation and code generation features make it suitable for rapid prototyping and embedded system development.



Simulations: Mission Models

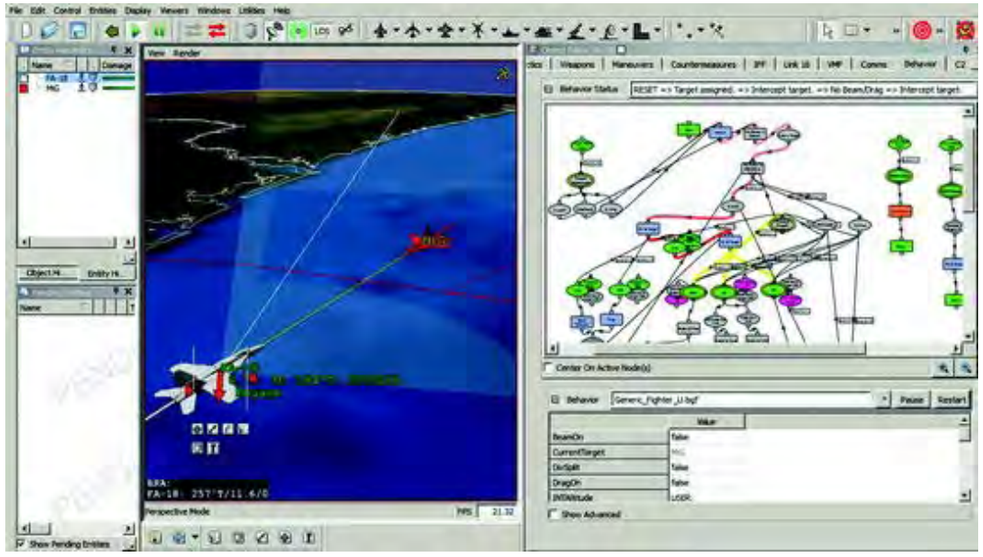
Mission models are specialized simulation tools that replicate entire operational environments or complex, multi-system interactions. These simulations can represent how various systems perform together in real-world scenarios, such as a satellite deployment or a coordinated drone operation. Mission models help engineers evaluate overall system performance, mission success rates, and potential points of failure across interconnected components and environments, ensuring readiness and reliability before deployment. Here are examples of two Mission Model tools:

AFSIM

AFSIM (Advanced Framework for Simulation, Integration, and Modeling) is a government-owned, open-source military simulation framework developed by the U.S. Air Force Research Laboratory (AFRL). It enables engineers to construct and analyze complex, multi-domain scenarios encompassing air, space, land, sea, and cyber operations. AFSIM supports the assessment of new system concepts, evaluation of tactics, and mission effectiveness analysis. Its modular architecture allows for the integration of custom models and behaviors, facilitating detailed simulations of platforms, sensors, weapons, and communication systems. This flexibility makes AFSIM a valuable tool for engineers involved in defense research, system development, and operational analysis.

NGTS (NEXT GENERATION THREAT SYSTEM)

NGTS (Next Generation Threat System) is a sophisticated military simulation system used to provide realistic training environments for pilots and other personnel by generating dynamic and evolving threat scenarios, allowing them to practice against a wide variety of simulated enemy aircraft, ground units, and naval vessels, all while receiving immediate feedback on their performance; it is primarily used by the US Navy and is integrated with existing training platforms to create an immersive experience.

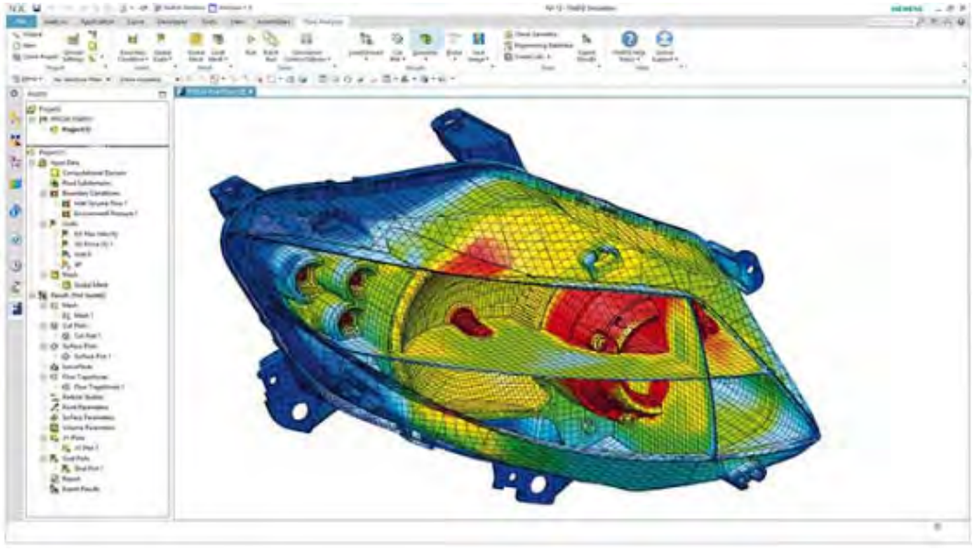


Simulations: Simulation Managers

Simulation Managers are a diverse set of simulation tools that provide a single interface for overseeing and orchestrating a set of other simulations, ensuring that data flows seamlessly between different tools and that results are tracked efficiently. These managers often include capabilities for automating simulations, collating data, and integrating findings into broader engineering workflows, supporting collaboration and traceability across engineering teams. Here are examples of a Simulation Manager:

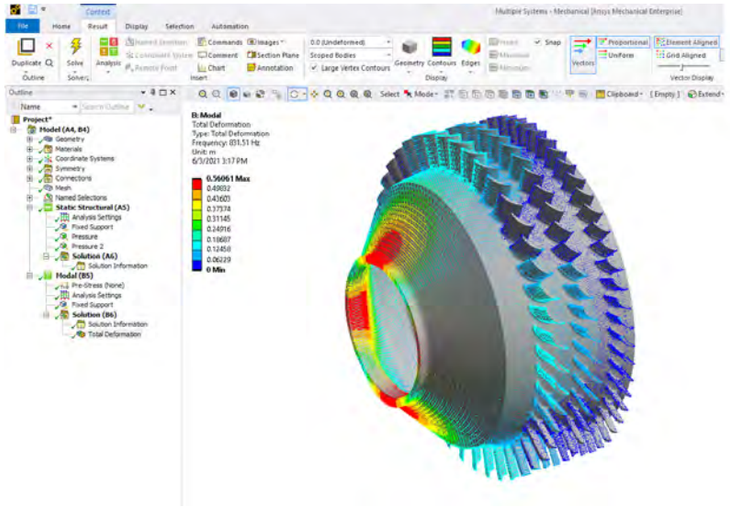
SIEMENS SIMCENTER

Simcenter is an advanced simulation and testing platform developed by Siemens Digital Industries Software that integrates multi-physics simulation, testing, and data analytics to enable comprehensive product performance analysis. It combines capabilities in computational fluid dynamics (CFD), finite element analysis (FEA), and system simulation to support end-to-end product development, from design validation to optimization. Simcenter's extensive toolset allows engineers to model and analyze complex interactions, such as thermal, structural, and acoustic phenomena, ensuring accurate performance predictions for intricate systems. With support for both 3D simulation and 1D system modeling, Simcenter enables teams to assess how individual components behave within broader systems, making it essential for industries like automotive, aerospace, and energy. This integrated approach helps accelerate the design cycle, improve product reliability, and support informed decision-making throughout the development process.



ANSYS MODELCENTER

Ansys ModelCenter is a robust integration and workflow automation software designed for model-based engineering, allowing engineers to link various simulation tools and models into cohesive, automated workflows. It enables the seamless orchestration of multi-disciplinary analysis, supporting efficient trade studies and optimization across diverse engineering domains. With ModelCenter, teams can integrate CAD, FEA, CFD, and other simulation software into a single workflow, automating data transfer and reducing manual effort. The platform's ability to handle complex simulations and connect disparate tools makes it invaluable for streamlining product development processes and improving collaboration in aerospace, automotive, and defense industries. By facilitating comprehensive design exploration and optimization, ModelCenter enhances decision-making and product performance throughout the engineering lifecycle.



Model Based System-Engineering (MBSE) and System Architecture and Design Tools

Systems engineering is a discipline that integrates various engineering fields and principles to design, manage, and optimize complex systems throughout their entire lifecycle. It focuses on defining customer needs, developing system requirements, and ensuring that all components work together harmoniously to achieve the overall objectives.

This discipline encompasses the coordination of different technical aspects, including system architecture, modeling, analysis, and verification, to ensure that the final product or system meets both functional and performance requirements, as well as balancing constraints like cost, schedule, and risk.

Systems engineering promotes a holistic approach that bridges technical specialties and supports effective communication and collaboration across multidisciplinary teams, and the set of System Architecture and Design Tools enable the broad deployment of standard and rigorous processes.

Requirements and compliance will always drive what engineers produce. To create a solution to a problem, they will always be limited by a number of boundaries and interdependencies that shape how they solve the problem. Developing any complex system requires that an engineer starts with a robust understanding of the business drivers, stakeholder needs, strategic goals, and human-factors/ergonomics, as well as regulatory and compliance standards could include environmental and safety requirements.

System Architecture and Design Tools are a set of software tools that help engineers and designers develop the structure and functional framework of complex systems to capture the requirements, the functions the system performs, the way that various subsystems interact, and how the physical components will solve the objectives.

Because the tools are designed to be used for complex systems they facilitate the creation, visualization, and analysis of a system architecture, to ensure that all the sub components and subsystems align with the overall system goals.

These tools hold the needs and goals of a complex system (for example, the overall objectives for creating a space station), the strategy for achieving those goals (such as defining the life-support, communication, and propulsion subsystems), and how those systems interact, as well as the integration of supporting systems (like thermal controls, power distribution, and data management). The tools ensure that each subsystem works cohesively, identifying relationships and dependencies between subsystems, and potential conflicts. These tools also outline verification and validation plans to help the engineer confirm that each subsystem meets its requirements, and the overall system functions as intended.

The engineer of the future needs to take this abstraction of the system and connect the dots so that the various engineers, such as a CAD engineer and a simulation engineer, can understand the objectives of the physical system, and how to ensure they don't miss a connection between what they are working on and what other engineers are creating. By connecting these disciplines together it creates a positive feedback loop to help evolve the architecture as simulations are completed, and ensure compliance to standards.

Using robust digital threading enables the engineer of the future to trace how a single piece of data might flow from a requirement to the functional, then logical, diagrams. And then to the physical diagrams and penultimately to the CAD design and simulations before ultimately being manufactured.

The engineer of the future will need to connect the output of their MBSE systems to other systems by creating comprehensive digital threads that enhance collaboration and lifecycle management. Some valuable data to be extracted from MBSE systems include:

- System Requirements and Traceability Data, linking requirements to models and other data sources for end-to-end visibility
- Architectural Models and Diagrams: Representations of system structure and behavior, supporting communication across teams
- Verification and Validation Plans: Documentation that aligns test cases with requirements and models to ensure compliance
- Simulation Data and Results: Outputs that show how systems perform under different scenarios, aiding risk management
- Change Management Records: Information on model updates and decision rationales for maintaining version control and traceability

These valuable data can take the form as:

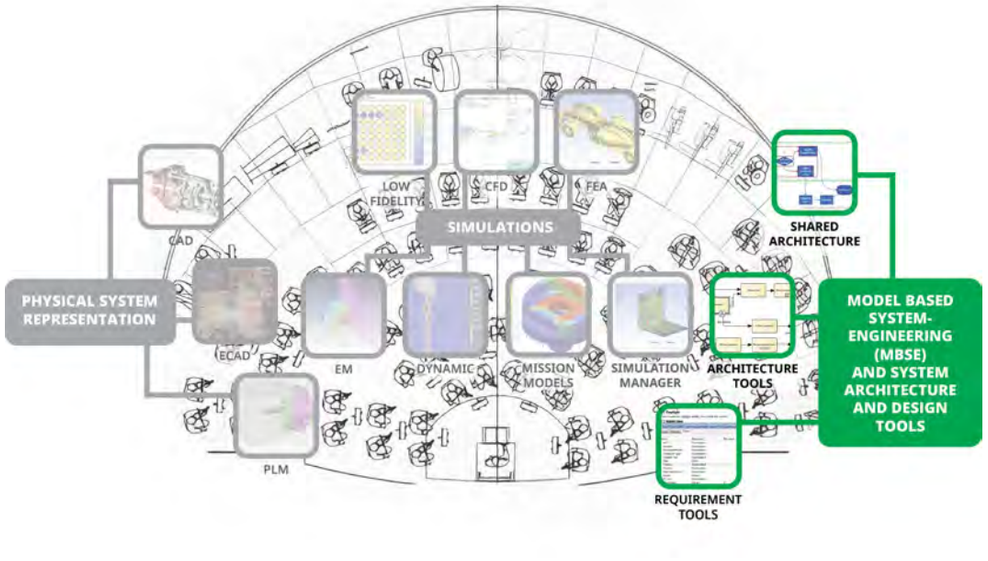
- Graphic images (such as .png) of all types of diagrams
- Structured data (such as .json) of diagrams such as body block diagrams and requirements
- Structured data in spreadsheet form (such as .csv) from all tables like Requirements tables

Supporting this process are a set of tools including:

- **Requirements Management Tools:** used for capturing, organizing, prioritizing, and tracking all project requirements
- **Architecture Tools:** used to define and visualize the structure and organization of complex systems
- **Shared Architecture Tools:** used for collaboration across teams and projects

These tools enable engineers to capture requirements, define system architecture, and create a shared framework that supports decision-making and development. Below are the key sub-disciplines of MBSE and system architecture tools.

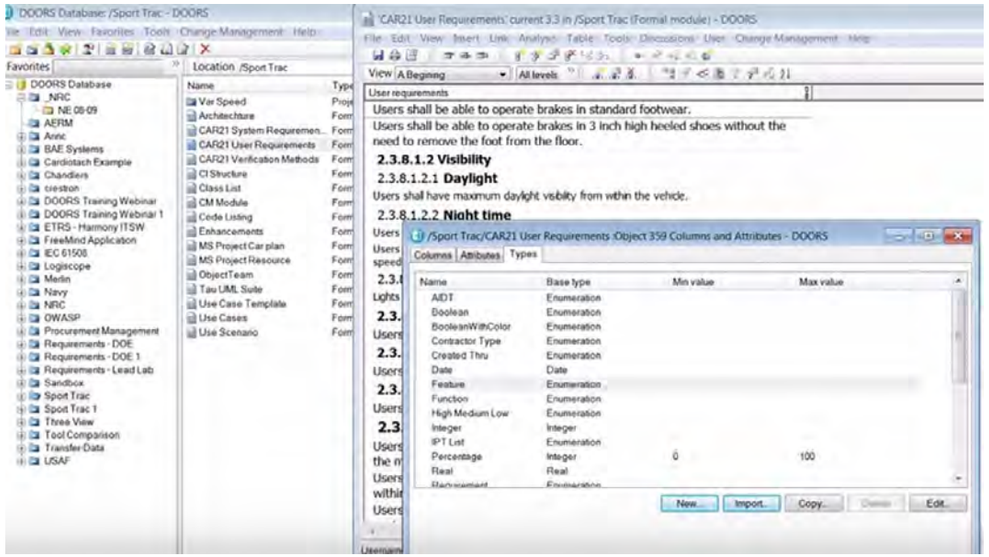
Model Based System-Engineering: Requirements Tools



At the core of any successful engineering project is the precise definition and management of requirements. Requirements management tools facilitate the process of capturing, organizing, prioritizing, and tracking all project requirements from initial concept to final deployment. These tools ensure that stakeholders' needs, compliance standards, and project goals are clearly documented and accessible. By maintaining a continuous traceability chain, engineers can align design efforts with requirements throughout the development lifecycle, reducing the risk of unmet expectations, scope creep, or design misalignment. This alignment helps teams verify that each element of the system meets its intended function, enhancing quality and reducing rework. Here are some examples of popular MBSE Requirements Tools:

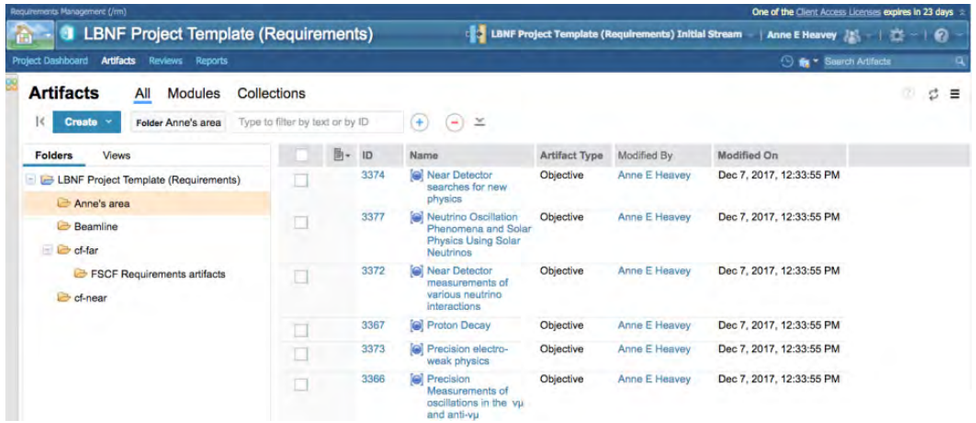
IBM DOORS

IBM Engineering Requirements Management DOORS which stands for (Dynamic Object-Oriented Requirements System) is a tool designed to capture, trace, analyze, and manage requirements for complex systems and software development. It enables users to store multiple types of requirements in a common repository, facilitating effective management in large projects. DOORS integrates with other products to provide a complete set of applications for software or systems development. A new version of this tool called DOORS Next enabled web-based functionality on top of the trusted DOORS features.



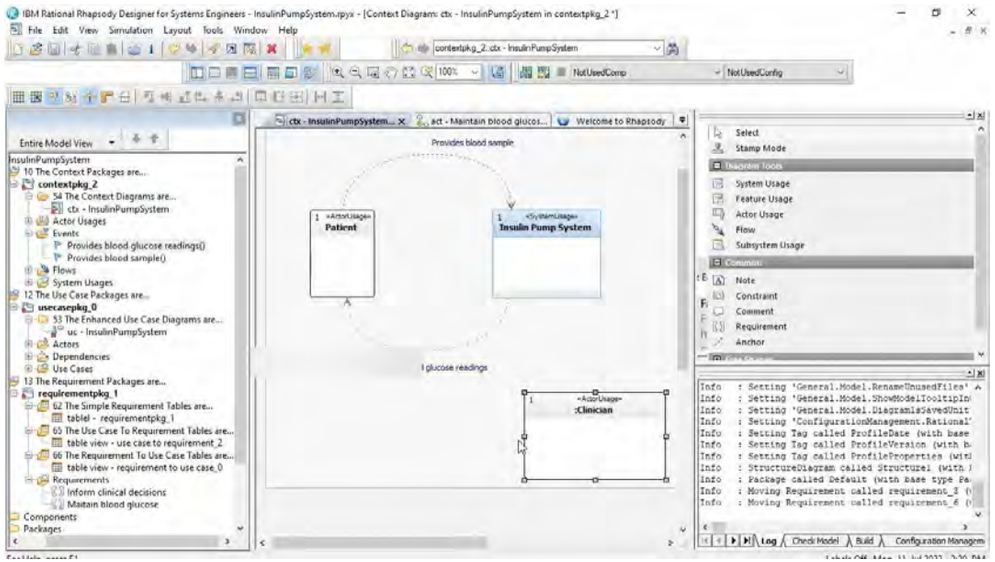
IBM DOORS NG

IBM DOORS Next Generation (DOORS NG) is a more modern web-based version of the DOORS requirements management tool, also built for complex engineering projects that require detailed traceability and team collaboration. Unlike the traditional IBM DOORS, which is client-based and known for its complex user interface, DOORS NG is web-based, offering a more modern and user-friendly experience as part of the IBM Engineering Lifecycle Management suite. It supports comprehensive requirements capture, analysis, and documentation, facilitating effective change management throughout the project lifecycle. DOORS NG's integration capabilities allow real-time collaboration among stakeholders and ensure that requirements are traceable through design, development, and testing phases, supporting adherence to industry standards. This positions DOORS NG as a practical option for industries like aerospace, defense, and automotive that need detailed oversight of complex requirements.



IBM RATIONAL RHAPSODY

IBM Rational Rhapsody is a modeling environment designed for MBSE that supports the development of complex systems through UML and SysML languages. It enables systems engineers and developers to design, simulate, and validate system behavior, providing tools for requirements traceability, code generation, and real-time modeling. Rhapsody helps streamline collaboration among engineering teams by integrating with other tools and allowing continuous validation of requirements throughout the development lifecycle.

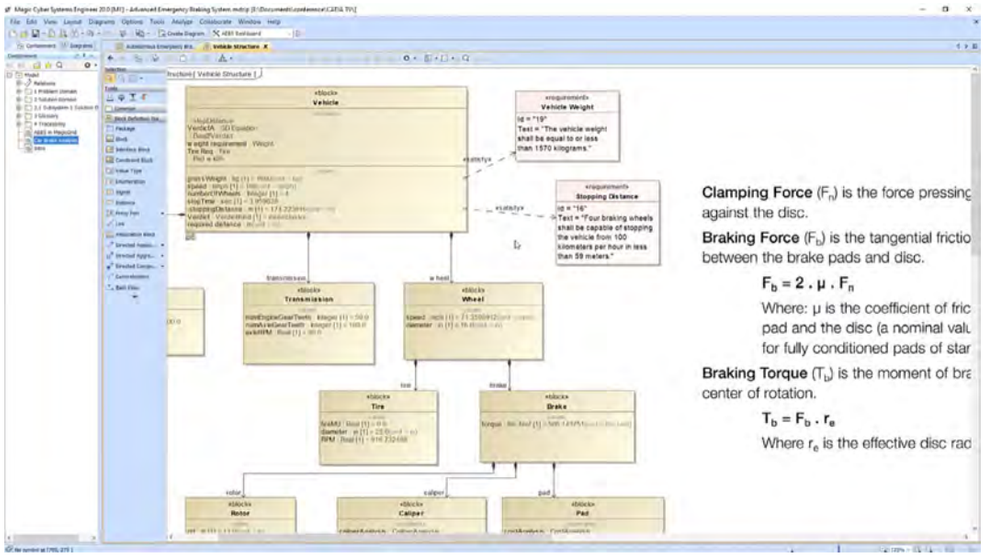


Model Based System-Engineering: Architecture Tools

Architecture tools are essential for defining and visualizing the structure and organization of complex systems. These tools help engineers model how various components and subsystems interact within a system, detailing logical and physical relationships, data flows, and system hierarchies. By providing a clear overview of system architecture, these tools allow engineers to plan integration points, ensure modularity, and maintain consistency across different development stages. Architecture tools often leverage visual modeling languages like SysML (Systems Modeling Language) to create structured, comprehensive diagrams that effectively communicate system design to teams and stakeholders, supporting better decision-making and efficient project development. Here are some examples of popular tools used for MBSE Architecture:

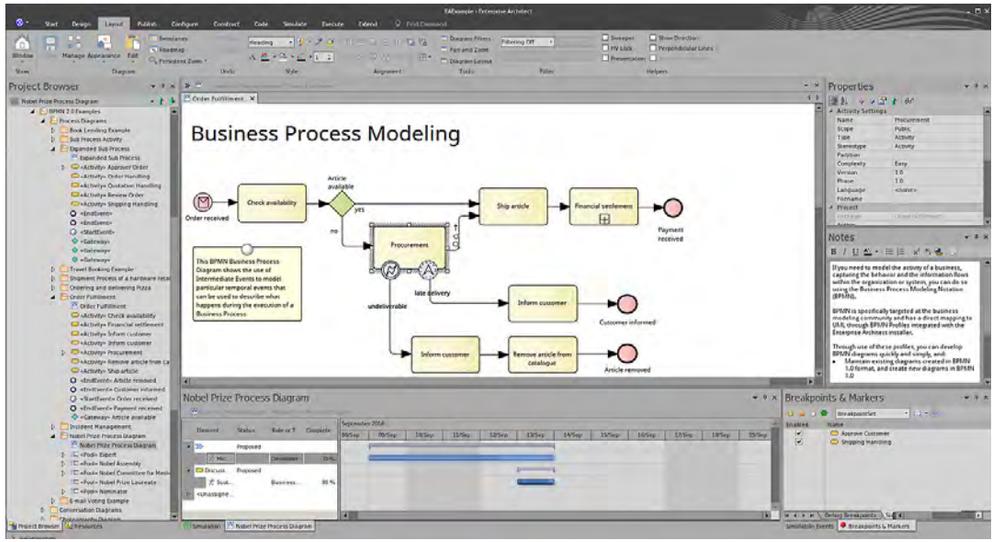
DASSAULT CAMEO

Dassault Cameo Systems Modeler or Cameo Enterprise Architect are cross-platform, collaborative environments tailored for MBSE. It provides robust tools to define, track, and visualize all aspects of systems through standard-compliant SysML models and diagrams. This enables systems engineers to perform engineering analyses for design decision evaluation and requirements verification, ensuring consistency and traceability throughout the system development lifecycle.



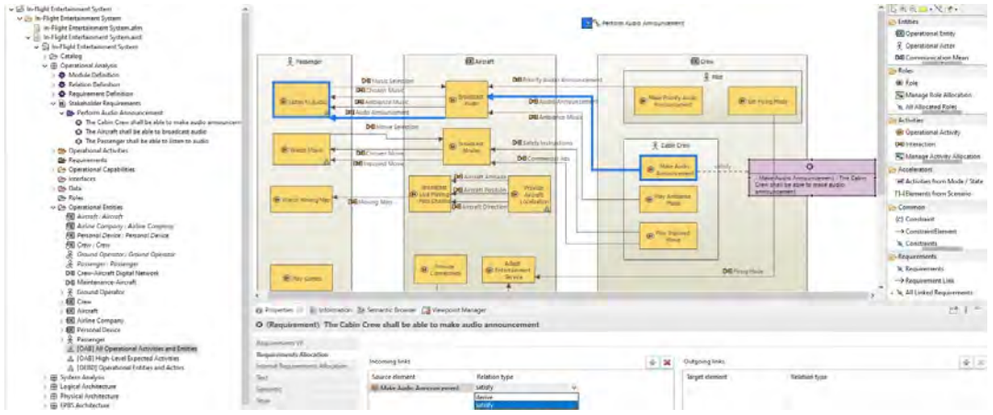
SPARX EA (ENTERPRISE ARCHITECT)

Sparx EA (Enterprise Architect) is a comprehensive modeling tool used for enterprise architecture, systems engineering, and software development across various industries. It provides robust support for visualizing, analyzing, and managing complex system architectures through detailed models and diagrams. The platform facilitates effective collaboration among teams by ensuring that system requirements, design specifications, and business processes are captured accurately and maintained consistently. With features such as traceability, impact analysis, and multi-layered modeling, Sparx EA helps organizations align system design with strategic business objectives and adapt efficiently to evolving project needs.



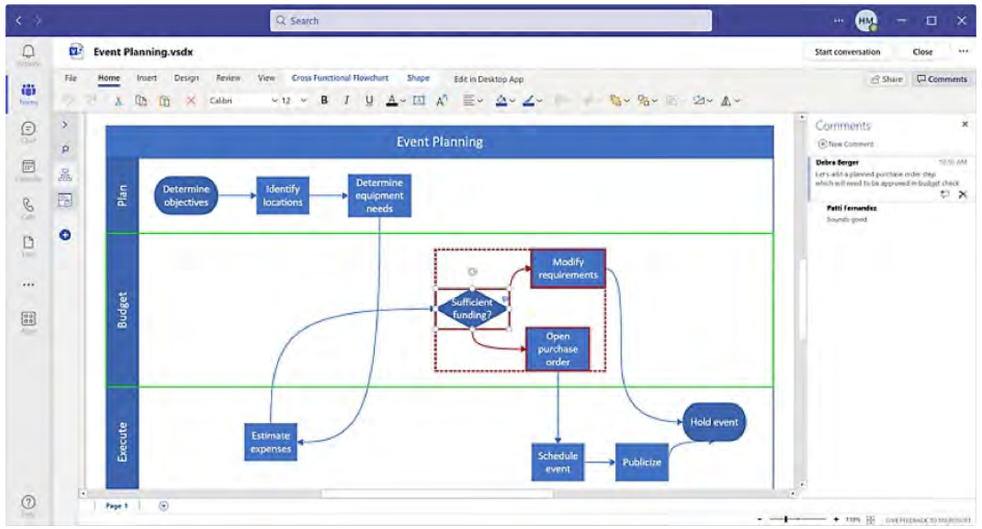
ECLIPSE CAPELLA

Eclipse Capella is an open-source model-based systems engineering (MBSE) tool designed to support the design and analysis of complex systems. Developed as part of the Eclipse ecosystem, Capella provides a comprehensive approach for defining system architectures from high-level operational analysis down to detailed design. It includes features for requirements management, functional analysis, and physical architecture design, enabling teams to ensure consistency and traceability across different stages of system development. Capella is particularly noted for its user-friendly interface and its ability to visualize interactions between system components, making it a valuable tool for engineering projects in industries such as aerospace, automotive, and defense. Through its extensibility and integration with other Eclipse tools, Capella facilitates collaboration and adaptability in model-based engineering workflows.



MICROSOFT VISIO

Microsoft Visio diagramming tool that allows MBSE users to create flowcharts, system diagrams, and process maps, providing a visual means to capture and communicate systems engineering concepts. While not specifically tailored for MBSE, Visio supports the creation of UML and basic SysML diagrams that can help with initial modeling and communication of system designs. It is often used as a complementary tool for high-level overviews or when creating quick visual representations within MBSE projects.

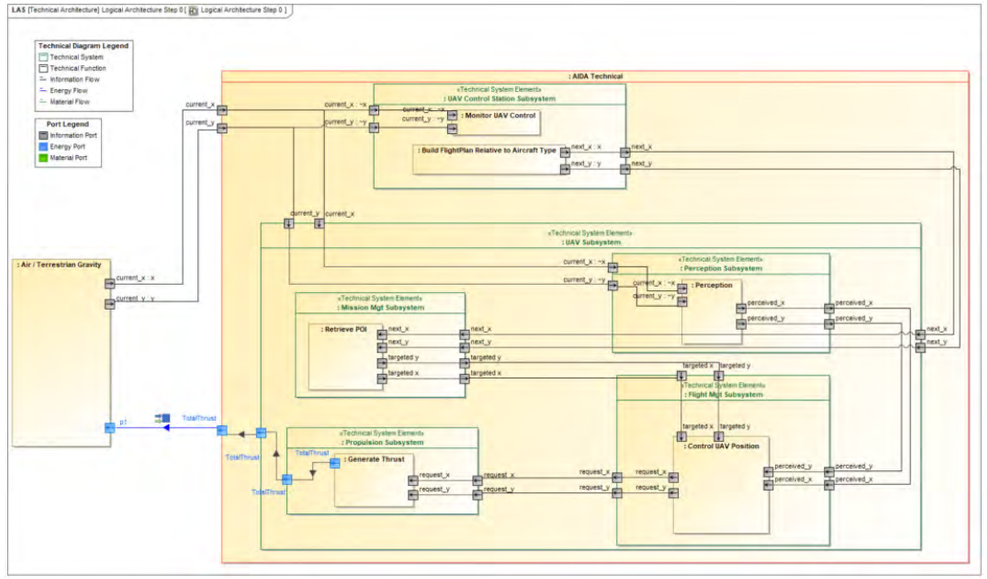


Model Based System-Engineering: Shared Architecture

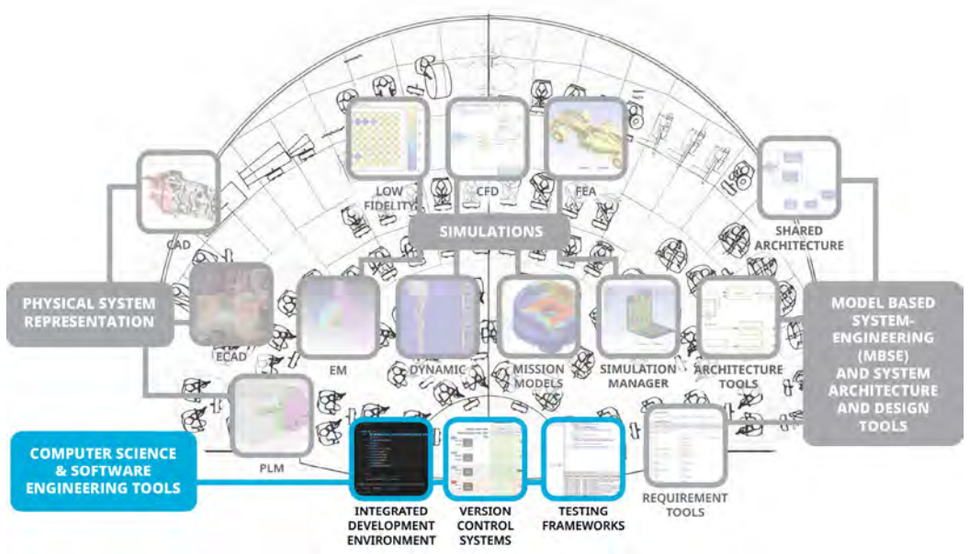
Shared architecture emphasizes collaboration across teams and projects by providing a common framework for system design. This sub-discipline focuses on creating and maintaining a unified source of truth with which multiple teams can reference and contribute. Shared architecture tools enable different teams to align their efforts under a standardized set of principles, components, and design patterns, ensuring that their work integrates seamlessly with other parts of the system. Here's an example of a Shared Architecture tool:

DASSAULT TEAMWORK CLOUD

Dassault Teamwork Cloud is a central repository designed for collaborative development and versioned storage of models in Model-Based Systems Engineering (MBSE) most commonly for Cameo models. It enables team members to access and modify the same model or diagram simultaneously, facilitating efficient collaboration. Key features include role-based access control, element-level versioning, and a web-based administrative interface for managing user accounts, licensing, and access control. Additionally, Teamwork Cloud supports integration with third-party tools, enhancing its utility in complex engineering environments.



Computer Science and Software Engineering Tools



Software development tools have become the backbone of modern engineering projects, enabling complex system designs, simulations, and collaborations. Much like the impact that CAD and ECAD tools have had on revolutionizing mechanical and electronic design, software engineering tools like Integrated Development Environments (IDEs), version control systems, and continuous integration platforms have transformed how engineers develop, test, and maintain software components integral.

The roots of software engineering tools can be traced back to the early days of computing, where programming was performed using punch cards and assembly language. As software complexity grew, so did the need for better tools to manage code, collaborate among teams, and automate repetitive tasks. The 1970s and 1980s saw the emergence of text editors and basic compilers, but it wasn't until the 1990s and 2000s that IDEs like Eclipse and Visual Studio provided integrated environments for coding, debugging, and testing.

Version control systems evolved from simple file locking mechanisms to sophisticated distributed systems like Git, introduced by Linus Torvalds in 2005. These tools have become indispensable for managing large codebases and coordinating work among distributed teams. Similarly, continuous integration tools like Jenkins emerged to automate the building, testing, and deployment of software, ensuring that code changes integrate smoothly and systems remain stable.

Today's software engineering tools are highly advanced, offering features that enhance productivity, collaboration, and code quality:

- **Integrated Development Environments (IDEs):** Modern IDEs like Visual Studio Code (VS Code) provide intelligent code completion, real-time error checking, debugging tools, and plugins for a multitude of programming languages and frameworks. They serve as a one-stop-shop for developers, integrating various aspects of software development into a single interface.
- **Version Control Systems:** Git has become the de facto standard for version control, enabling developers to track changes, revert to previous states, and collaborate efficiently. Platforms like GitHub and GitLab add layers of project management, issue tracking, and social coding, fostering community and collaboration.
- **Continuous Integration/Continuous Deployment (CI/CD):** Tools like Jenkins automate the process of building, testing, and deploying code. They help catch issues early in the development cycle, reduce manual errors, and speed up the release process.

In our evolving digital landscape, the engineer of the future must be as comfortable writing code as they are designing physical components. Software has become the glue that binds different engineering disciplines, enabling systems to communicate, adapt, and learn. Whether it's developing embedded software for a new piece of hardware, creating automation scripts to streamline workflows, or leveraging AI algorithms to optimize designs, software engineering skills are indispensable.

Modern software interfaces enable the developer to provide a set of instructions to a tool and view the results. Over the evolution of music, musicians evolved their ability to notate their set of instructions for their tools: the instruments. Early music notation would indicate notes,

but not the duration of each note. Later notation helped the viewer understand how long a note is played, and as the instruments developed so did notation for creating various dynamics. For example the violin can create many different sounds, from beautiful ones used in symphonies written in the 1800s to the eerie sounds in horror movies that famously include the violin sounds used in Psycho and The Shining. In order to communicate these intents, a composer has methods to notate many ways a violinist should play. They can indicate the bow direction (up or down), if the musician should play very close to the bridge (for a glassy or metallic sound), further down the strings-above where the black fingerboard ends but before the f-shaped holes are (for a soft sound), or if they should strike the string with the bow and a short movement to create a loud sharp sound.

Much like our composer, our engineer of the future needs to learn the skill of software coding in order to more expressively connect various digital engineering tools in meaningful ways so they can be used more dynamically. All digital engineering tool manufacturers create useful GUIs which enable someone with little training to manipulate the data in specified workflows. These tool manufacturers know that the user interface can't be optimized for every scenario so they also create an API to enable software developers to reach through the tool to get access to the functions of the tool. The engineer of the future will write code to use these APIs in order to create robust digital threads to connect all their tools in ways that allow each tool to express what it does best, as well as how and when it performs the actions.

The engineer of the future will need to:

- **Understand Programming Fundamentals:** Grasp basic coding concepts and be able to read and write code, even if it's not their primary role.
- **Leverage Automation:** Use scripts and tools to automate repetitive tasks, improving efficiency and reducing errors.
- **Collaborate Using Version Control:** Work effectively in teams using Git, managing branches, merging changes, and resolving conflicts.
- **Integrate Continuous Practices:** Embrace CI/CD pipelines to ensure that their contributions integrate smoothly with the larger system.

To build robust digital threads that connect software development with other engineering domains, it's crucial to manage and extract valuable data:

- **Source Code Repositories:** Version history, commit messages, and code changes that provide insights into the development process.
- **Build Artifacts:** Compiled binaries, libraries, and executables that are the outputs of the build process.
- **Test Results:** Data from automated test suites, including unit tests, integration tests, and system tests.
- **Issue Tracking Data:** Records of bugs, feature requests, and tasks that help manage project progress.
- **Documentation:** Code comments, README files, and API documentation that aid in understanding and maintaining the software.
- **CI/CD Pipeline Configurations:** Scripts and configurations that define how software is built, tested, and deployed.
- **Dependency Management:** Information about libraries and external dependencies required for the software to function.

As products become more interconnected and software-driven, the lines between software and traditional engineering disciplines will continue to blur. Engineers will need to adopt a holistic approach, understanding how software impacts and integrates with mechanical, electrical, and system designs.

Emerging trends like artificial intelligence, machine learning, and the Internet of Things (IoT) will further elevate the importance of software engineering tools. Engineers will leverage AI-assisted coding, predictive analytics, and smart automation to design more complex and adaptive systems.

Embracing these tools and practices will enable the engineer of the future to not only keep pace with technological advancements but also drive innovation, efficiency, and collaboration in the digital engineering landscape.

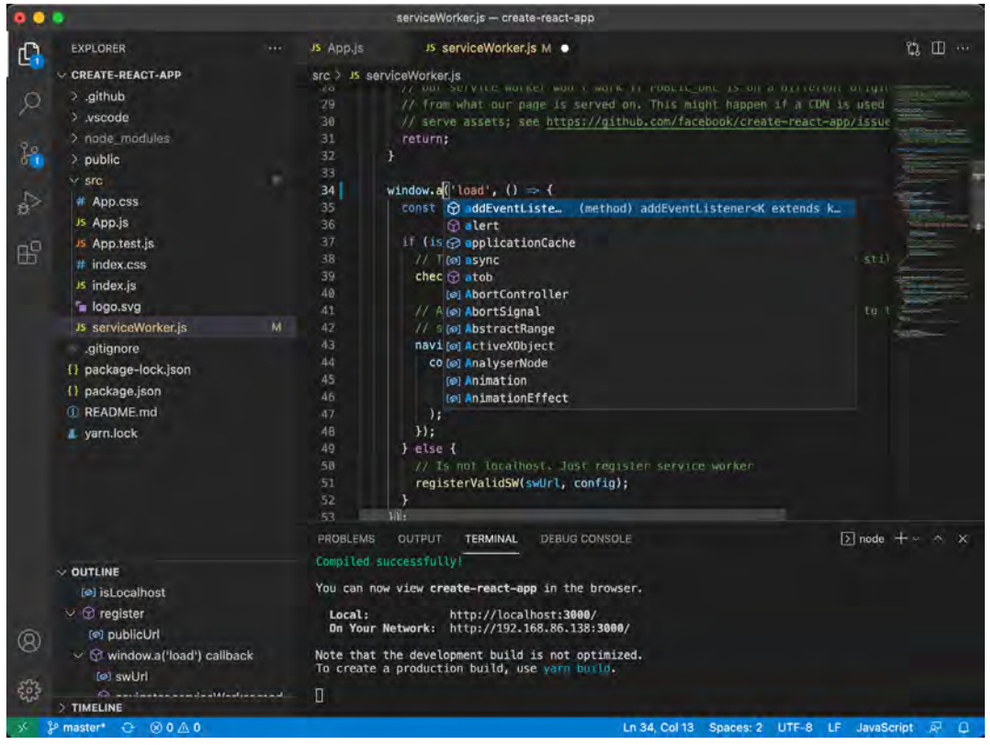
Let's explore each area in more depth.

Software Engineering: Integrated Development Environment (IDE)

An Integrated Development Environment is a comprehensive software suite that combines several tools used for software development into a single, cohesive user interface. These tools typically include a code editor, compiler or interpreter, debugger, and build automation features. IDEs are designed to streamline the coding process by offering syntax highlighting, code completion, and real-time error detection, enhancing productivity and minimizing errors. Here is an example of a popular IDE:

VISUAL STUDIO CODE (VS CODE)

Visual Studio Code (VS Code): A lightweight yet powerful source code editor developed by Microsoft. It supports a wide range of programming languages and comes with features like intelligent code completion (IntelliSense), debugging, and Git integration. VS Code's extensibility through plugins makes it adaptable to various development needs.



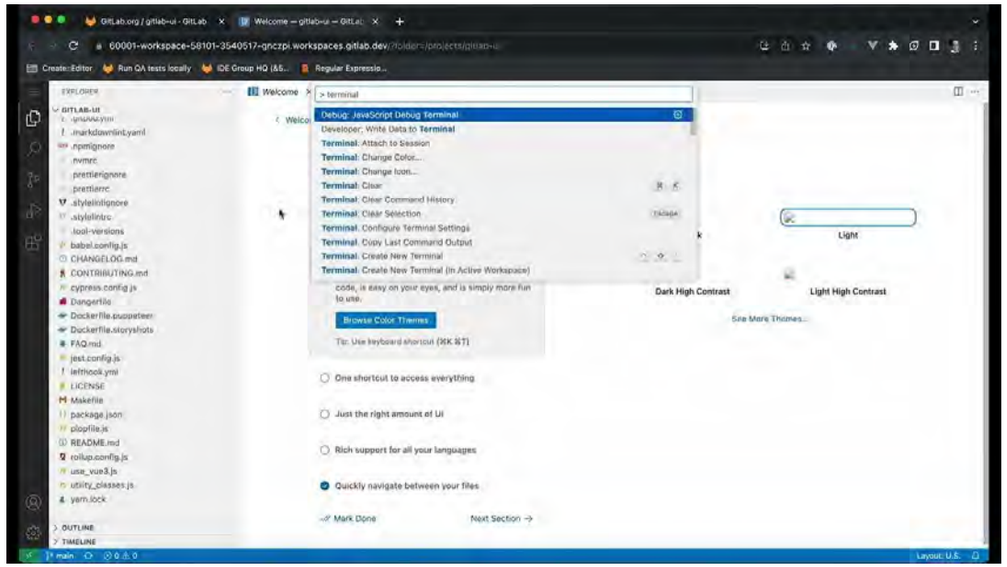
Software Engineering: Version Control Systems and Continuous Integration/ Continuous Deployment (CI/CD)

Version Control Systems are essential for tracking and managing changes to software code over time. They enable multiple developers to work collaboratively on a project without overwriting each other's work. VCSs maintain a history of code changes, allowing developers to revert to previous versions, track the origin of specific changes, and identify who made modifications. This ensures project continuity and reduces risks associated with development errors. Here are some examples of popular Version Control Systems:

GIT, GITLAB, GITHUB

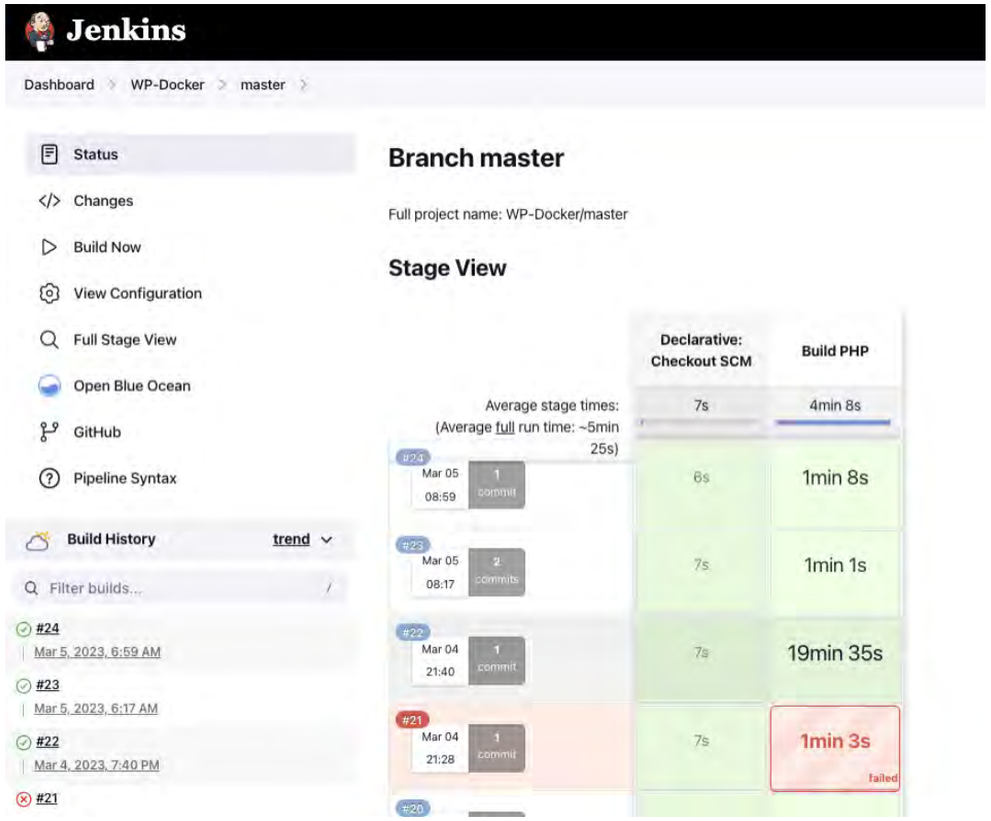
Git is a distributed version control system that allows multiple developers to work on the same codebase simultaneously. Git tracks changes, supports branching and merging, and helps resolve conflicts, making it essential for collaborative projects.

GitLab and GitHub: Web-based platforms built around Git that provide repository hosting, issue tracking, continuous integration, and deployment tools. They enhance collaboration through features like pull requests, code reviews, and project management boards.



JENKINS

Jenkins is an open-source automation server that enables developers to build, test, and deploy applications automatically. Jenkins supports a vast ecosystem of plugins, allowing it to integrate with numerous tools and technologies.

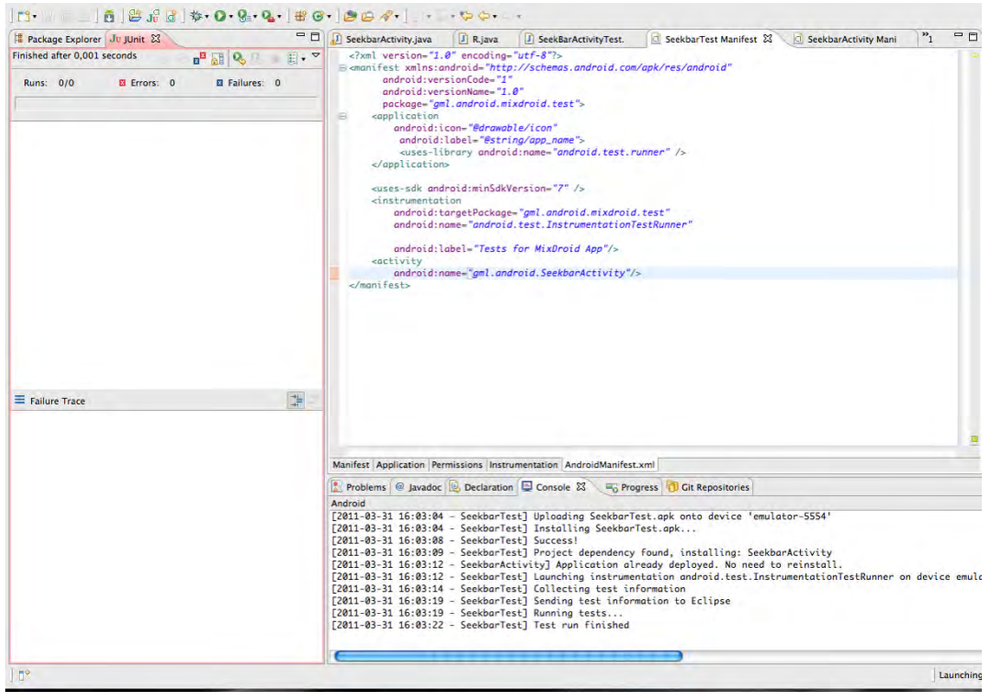


Software Engineering: Testing Frameworks

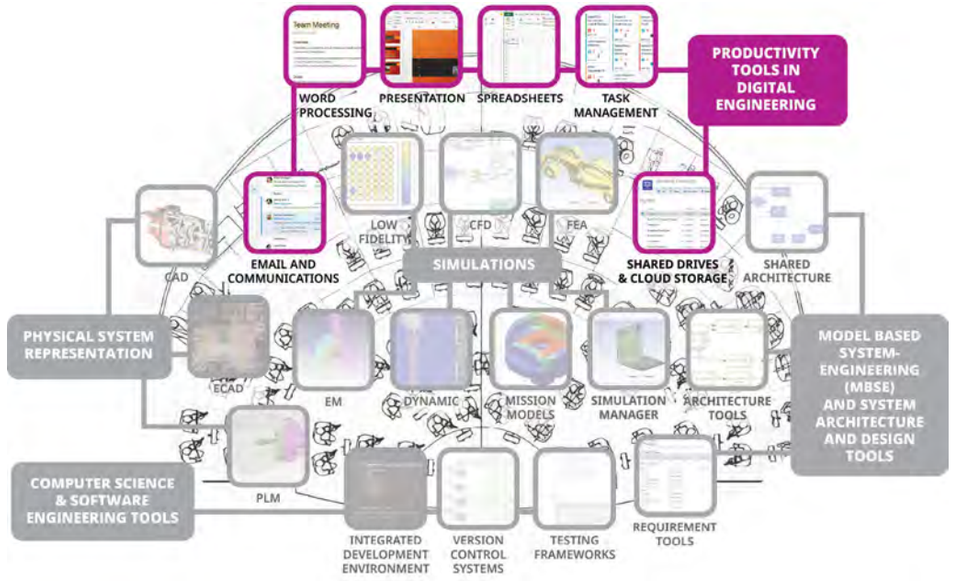
Testing frameworks are tools that help automate the process of verifying that code behaves as expected. These frameworks support different types of testing, including unit tests, integration tests, and functional tests, ensuring code reliability and reducing bugs. By automating tests, developers can quickly validate changes, catch potential issues early in the development cycle, and maintain software quality across updates. Here are examples of popular Testing Frameworks:

JUNIT, PYTEST, AND SELENIUM

JUnit, pytest, and Selenium are tools and frameworks that facilitate automated testing of software applications. They help ensure code quality by enabling developers to write and run tests that check for functionality, performance, and security issues.



Productivity Tools in Digital Engineering



The evolution of productivity tools mirrors the transformation of the modern workplace. We've gone from using typewriters to word processors, from filing cabinets to cloud storage, and from standalone calculators to comprehensive spreadsheet software.

As engineering projects grow in scale and complexity, the need has arisen for more sophisticated management and collaboration tools.

Today's productivity tools offer increasingly more powerful, interconnected, and user-friendly interfaces:

- **Word Processing and Document Collaboration:** Tools like Microsoft Word, Google Docs, and Apple Pages offer real-time collaboration, version control, and cloud integration. Multiple users can work on the same document simultaneously, regardless of their physical location.

- **Presentation Software:** Microsoft PowerPoint, Google Slides, and Keynote provide advanced features for creating engaging presentations, including multimedia integration, animations, and templates. They are essential for conveying complex engineering concepts to stakeholders.
- **Spreadsheets:** Microsoft Excel, Google Sheets, and Apple Numbers are used for data analysis, modeling, and visualization. They support complex calculations, statistical functions, and graphing tools that help engineers interpret data.
- **Email and Communication Platforms:** Outlook, Gmail, and Apple Mail remain staples for formal communication. However, instant messaging and collaboration platforms like Slack, Microsoft Teams, and even SMS or iMessage have become crucial for quick, informal exchanges.
- **Task Management and Project Planning:** Tools like Jira, Confluence, and Microsoft Project enable teams to plan projects, assign tasks, track progress, and manage resources. They provide visibility into workflows and help identify bottlenecks.
- **Shared Drives and Cloud Storage:** Google Drive, SharePoint, and Dropbox allow teams to store, share, and collaborate on files securely. They ensure that everyone has access to the latest versions of documents and that data is backed up and recoverable.

Extending our musical analogy, productivity tools are like the instrument used in some symphonies that everyone already has: the human voice. Everyone can sing, (not well, but they can sing), just like everyone can write an email (again, not everyone can write a good one). Productivity tools are so common that we sometimes forget we're even using them – and we don't often use them to their full potential and are unconsciously limited by these tools. Just as a group of strangers will rapidly join together to sing Happy Birthday, each in their own key (unfortunately), those same people would never think to sing 'Ode to Joy' from Beethoven's Symphony No.9 without practice. The engineer of the future will understand how to leverage these tools in new ways, to connect formal structured content in spreadsheets, to written content in word processors all the way through to their suite of digital engineering tools and back to presentation software that will help everyone at all levels of an organization easily understand the rich and complex data in simple and familiar formats.

KEY DATA TO EXTRACT FROM PRODUCTIVITY TOOLS

Integrating productivity tools into the digital thread enhances transparency, traceability, and collaboration. Key data includes:

- **Documentation:** Specifications, reports, meeting minutes, and project plans created in word processors.
- **Presentations:** Slide decks used to communicate designs, progress updates, and findings.
- **Data Analysis:** Spreadsheets containing calculations, models, budgets, and schedules.
- **Communication Logs:** Emails, messages, and discussion threads that may contain decisions, approvals, and critical information.
- **Task and Issue Tracking:** Data from project management tools that track tasks, deadlines, dependencies, and resource allocation.
- **Shared Files:** Documents, models, and other files stored in shared drives, ensuring everyone has access to the latest versions.

As engineering projects become more complex and teams more distributed, productivity tools will continue to evolve. Artificial intelligence and machine learning will enhance these tools, offering features like:

- **Automated Scheduling and Task Assignment:** AI can optimize project plans based on resource availability and priorities.
- **Intelligent Document Processing:** Natural language processing can help summarize documents, extract key information, and even draft content.
- **Enhanced Data Analysis:** Advanced analytics within spreadsheets can identify trends, anomalies, and insights more effectively.
- **Virtual and Augmented Reality Collaboration:** Emerging technologies may enable immersive meetings and collaborative design sessions, breaking down the barriers of physical distance.

The engineer of the future must be adaptable, continuously learning to leverage new tools and technologies to enhance productivity and collaboration. By effectively integrating these tools into their workflows, engineers can focus more on innovation and problem-solving, driving progress in the digital engineering landscape.

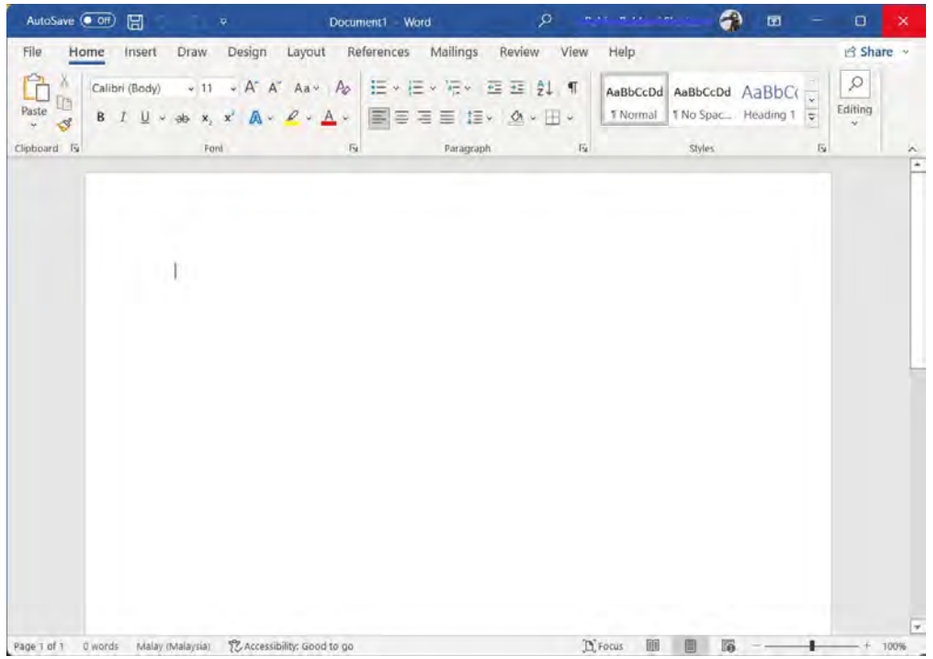
Let's explore the major areas of productivity software applications.

Productivity Tools: Word Processing

Word processing software has evolved significantly in recent years, now it's possible to collaborate with other people in near-real-time, link documents together between similar suites of tools, and leverage internet connectivity to add powerful AI functionality. Here are some popular word processing tools:

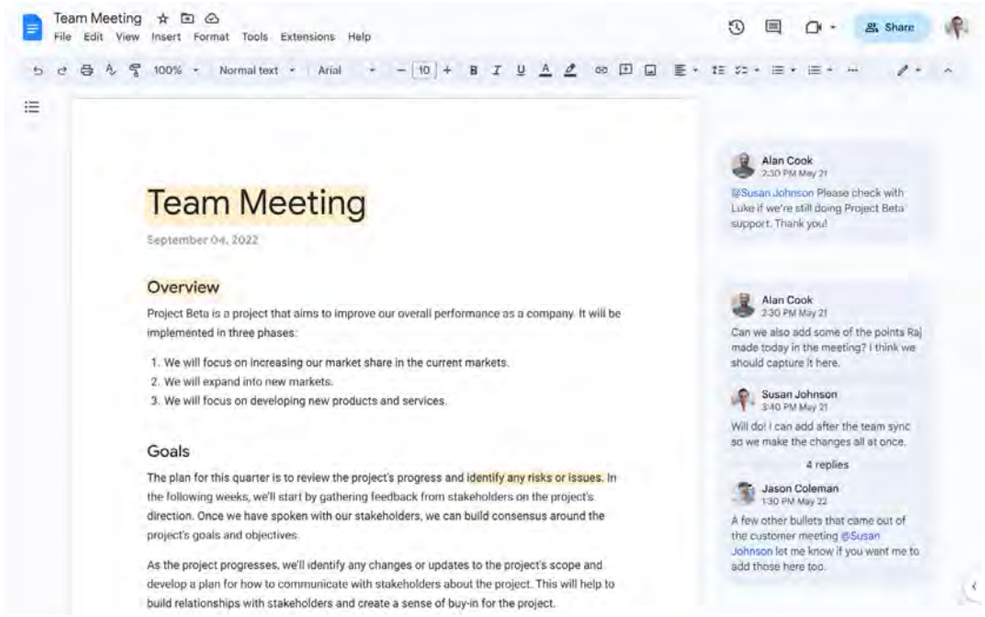
MICROSOFT WORD

Microsoft Word is a feature-rich word processor with robust formatting, collaboration, and review tools. Widely used in professional settings for creating formal documents.



GOOGLE DOCS

Google Docs is a cloud-based word processor that allows real-time collaboration and easy sharing. Accessible from any device with an internet connection.



Productivity Tools: Presentation

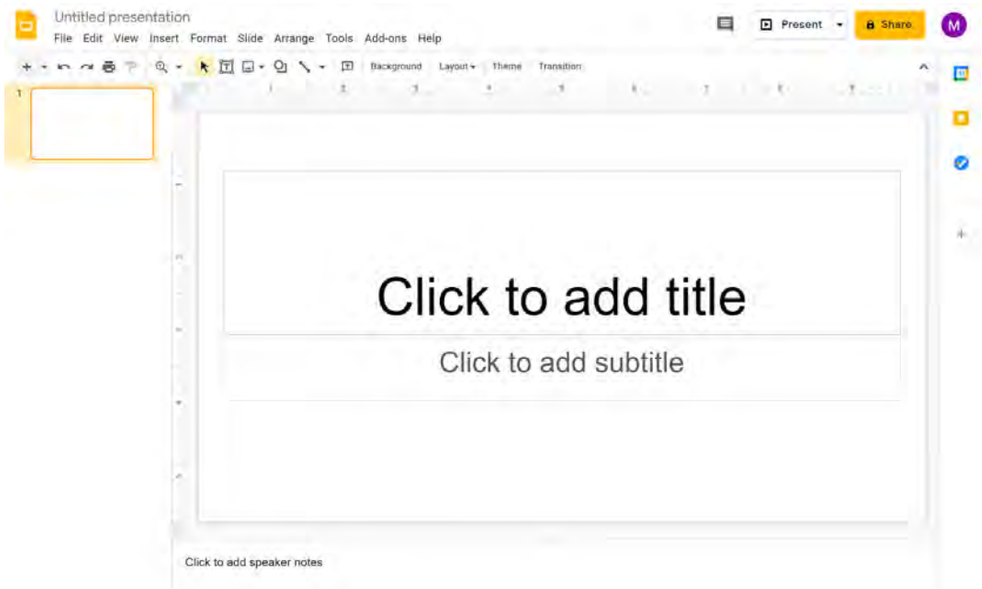
Presentation tools enable powerful storytelling when used properly, they can be used for people to quickly visualize ideas using simple and easy-to-edit shapes. Here are some popular presentation tools:

MICROSOFT POWERPOINT

Microsoft PowerPoint offers advanced features for creating professional presentations, including animations, transitions, and multimedia integration.

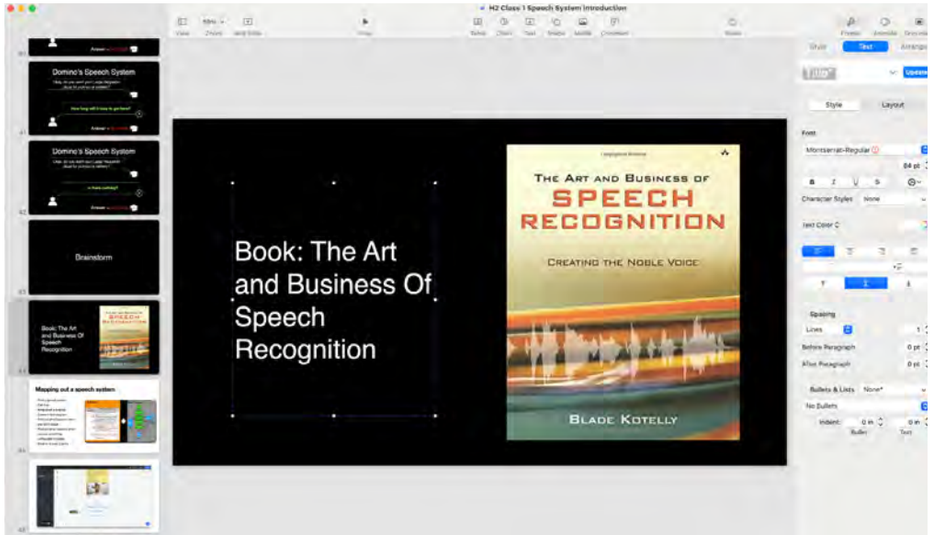
GOOGLE SLIDES

Google Slides enables collaborative presentation creation and sharing, with real-time editing and cloud storage.



APPLE KEYNOTE

Keynote is a presentation software developed by Apple, known for its intuitive design and powerful tools for creating visually engaging and professional presentations.

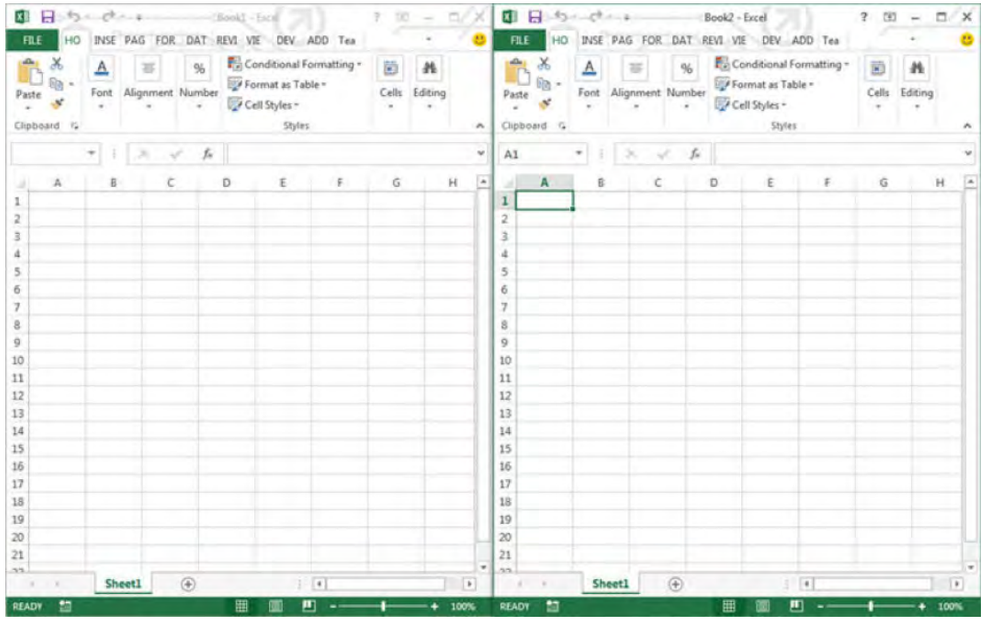


Productivity Tools: Spreadsheets

Spreadsheet tools facilitate efficient data organization, analysis, and visualization. When used properly, they empower users to create, manipulate, and present data clearly and effectively through numbers and graphs that update in real time. Some tools can be integrated into other systems providing robust static models. Here are some popular spreadsheet tools:

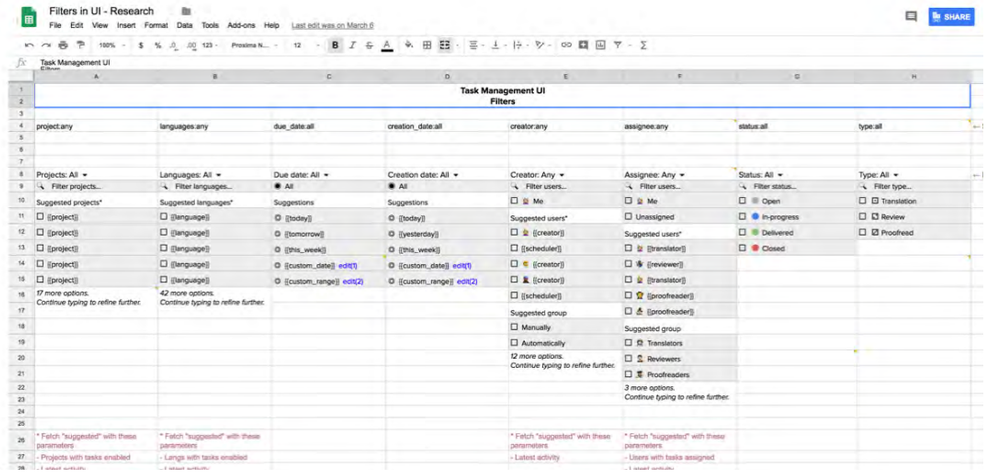
MICROSOFT EXCEL

Microsoft Excel is a powerful tool for data analysis, offering complex formulas, pivot tables, and data visualization capabilities.



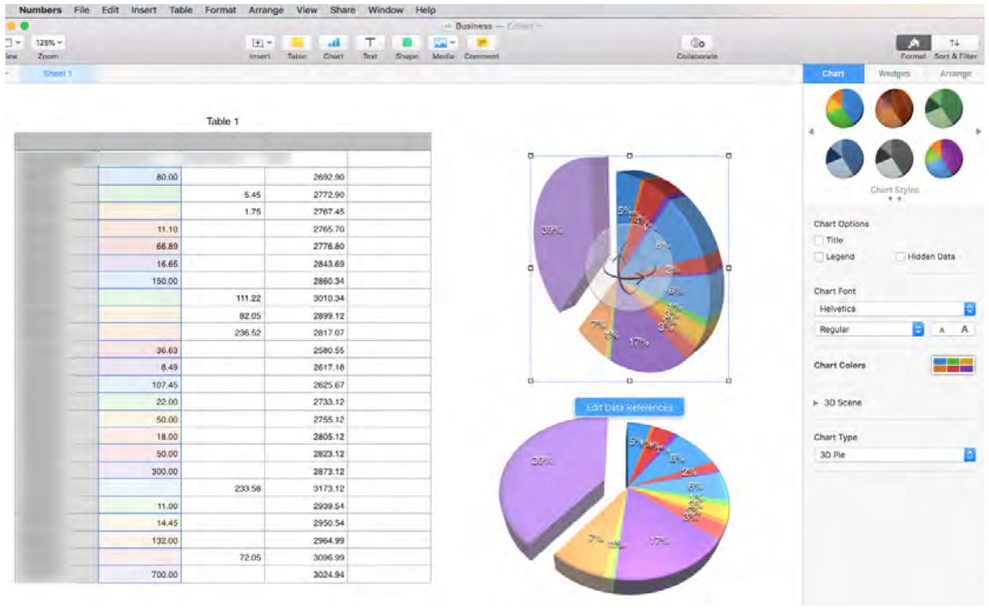
GOOGLE SHEETS

Google Sheets supports real-time collaboration on spreadsheets, with cloud-based storage and sharing.



APPLE NUMBERS

Apple Numbers offers an intuitive and visually appealing spreadsheet experience, with seamless integration across Apple devices and support for real-time collaboration through iCloud.

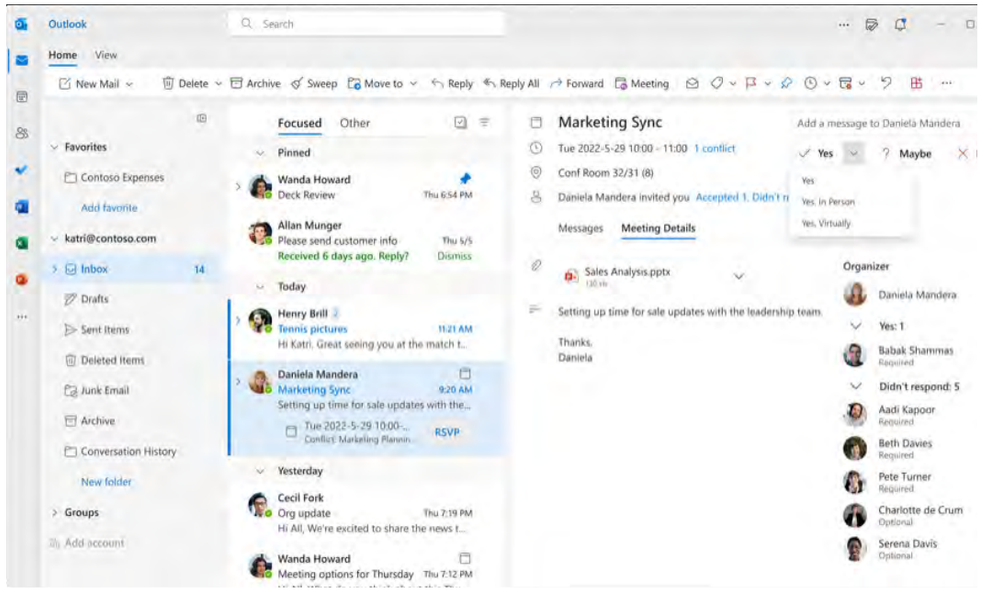


Productivity Tools: Email and Communications

Email and communication tools are essential for facilitating efficient and effective interaction within and outside organizations. When used properly, they help streamline workflow, enhance collaboration, and maintain clear communication channels. Here are some popular email and communication tools:

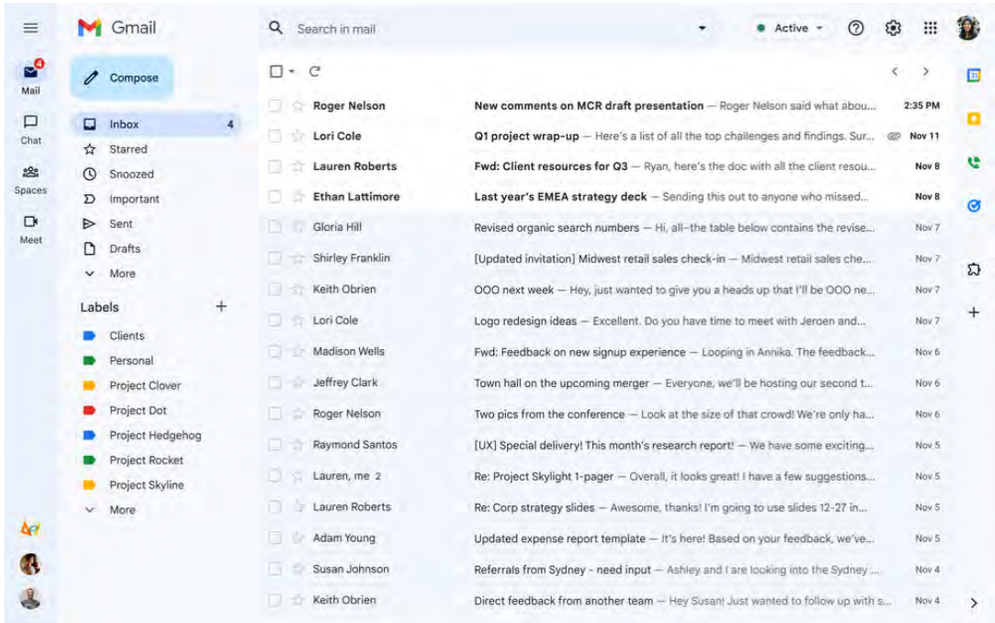
MICROSOFT OUTLOOK

Microsoft Outlook integrates email, calendar, and contacts, and is widely used in corporate environments.



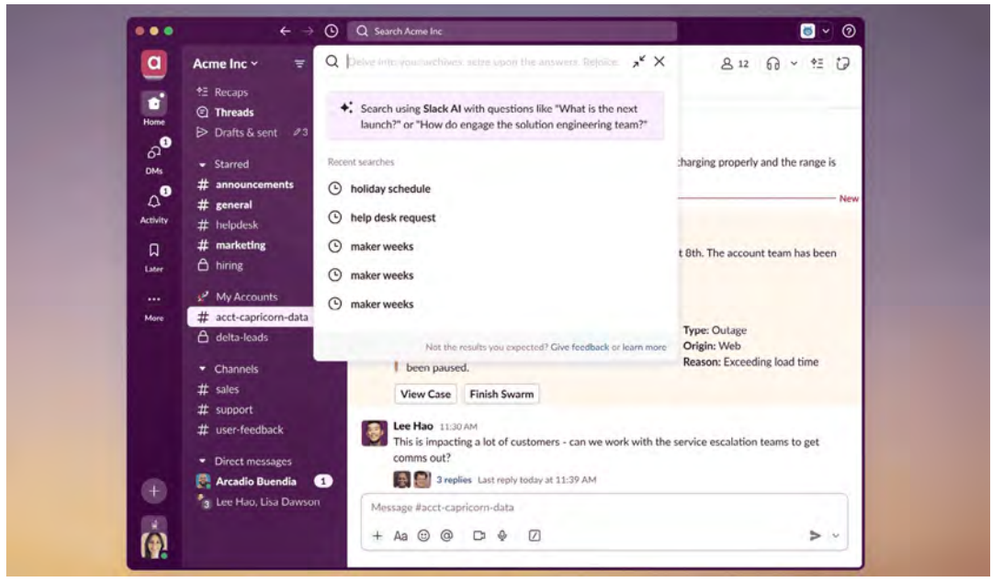
GMAIL

Gmail is a widely used email service by Google, offering a user-friendly interface, robust spam protection, and integration with other Google services for seamless communication and productivity.



SLACK

Slack is a messaging platform that supports channels, direct messaging, file sharing, and integrations with other productivity tools.

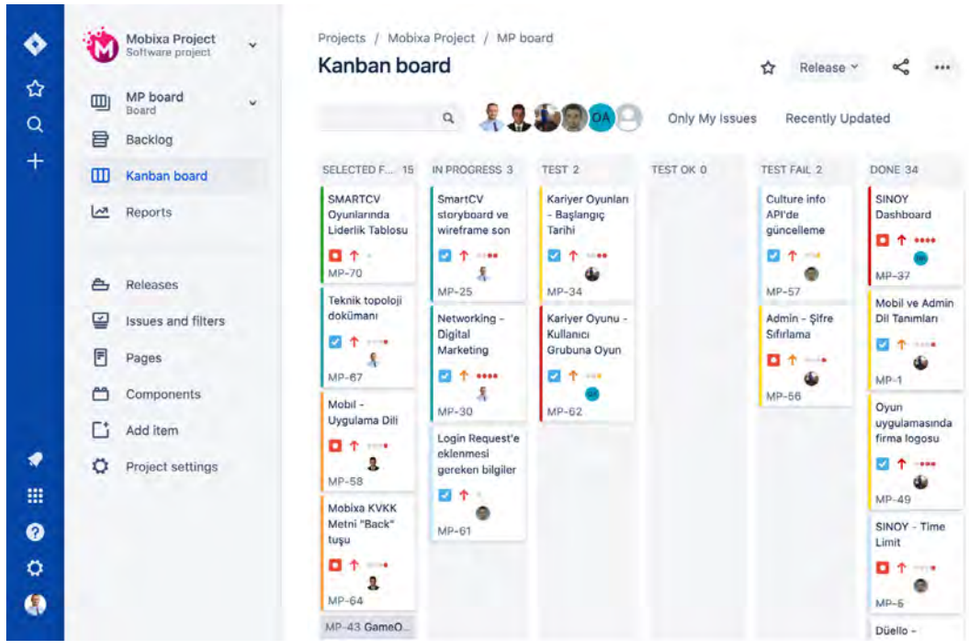


Productivity Tools: Task Management and Project Planning

Task management and project planning tools help teams and individuals organize their work, set priorities, track progress, and meet deadlines efficiently. When used effectively, these tools promote productivity, foster collaboration, and ensure that projects are completed successfully. Here are some popular task management and project planning tools:

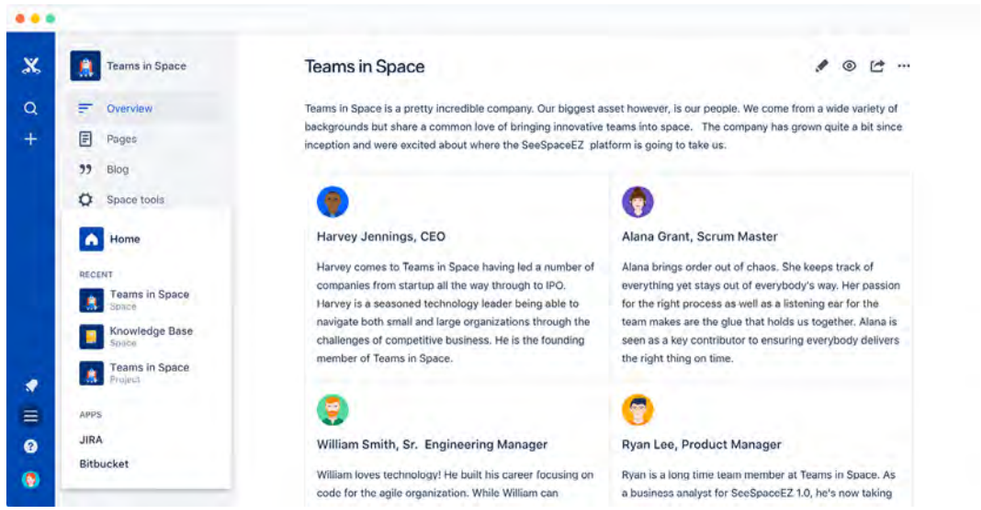
JIRA

Jira is a tool for issue tracking and agile project management, commonly used in software development but adaptable to other fields.



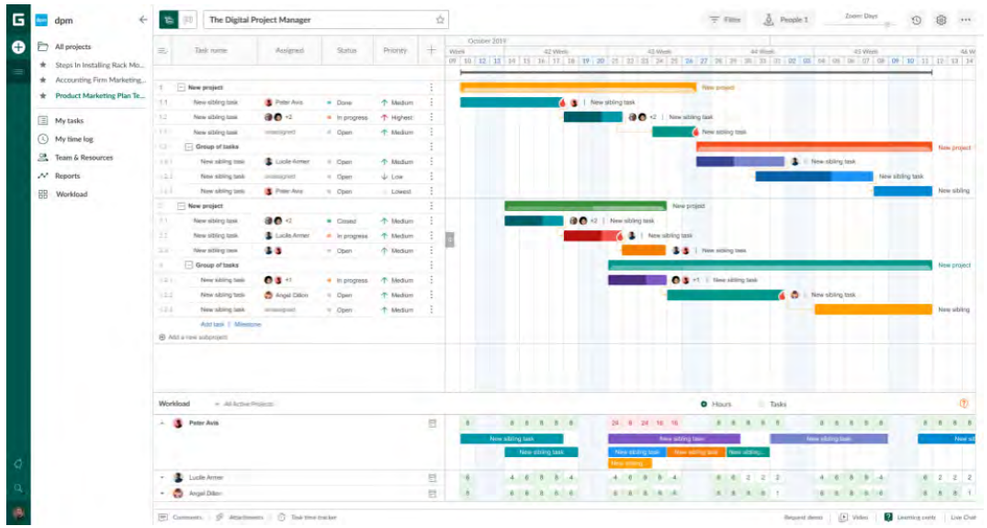
CONFLUENCE

Confluence is a collaboration wiki tool for sharing knowledge and documentation within teams.



MICROSOFT PROJECT

Microsoft Project provides features for project scheduling, resource management, and progress tracking.

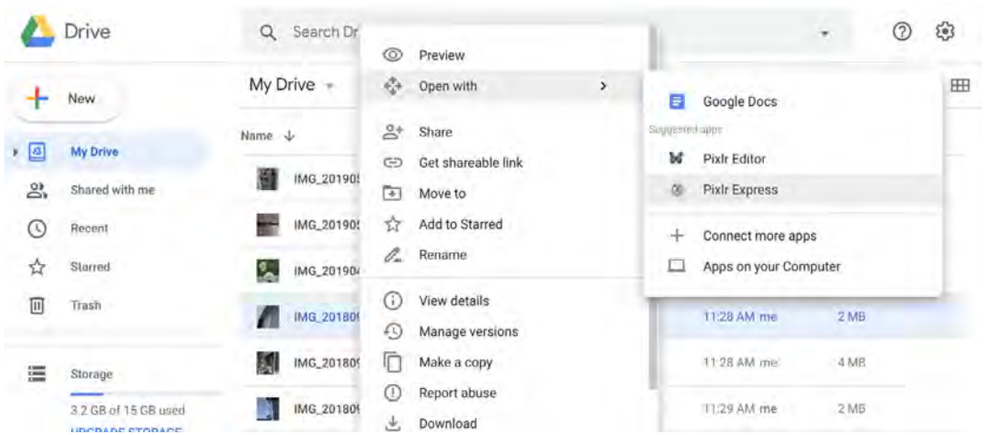


Productivity Tools: Shared Drives and Cloud Storage

Shared drives and cloud storage tools provide a secure and accessible way to store, manage, and share files across teams and organizations. When used effectively, these tools enhance collaboration, improve workflow efficiency, and ensure data is backed up and accessible from anywhere. Here are some popular shared drives and cloud storage tools:

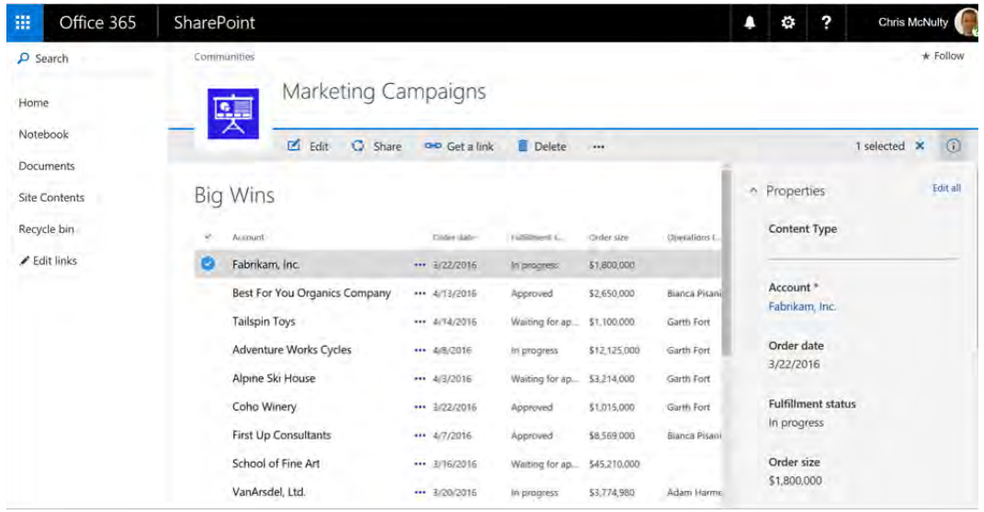
GOOGLE DRIVE

Google Drive offers cloud storage and synchronization, integrated with Google Workspace apps.



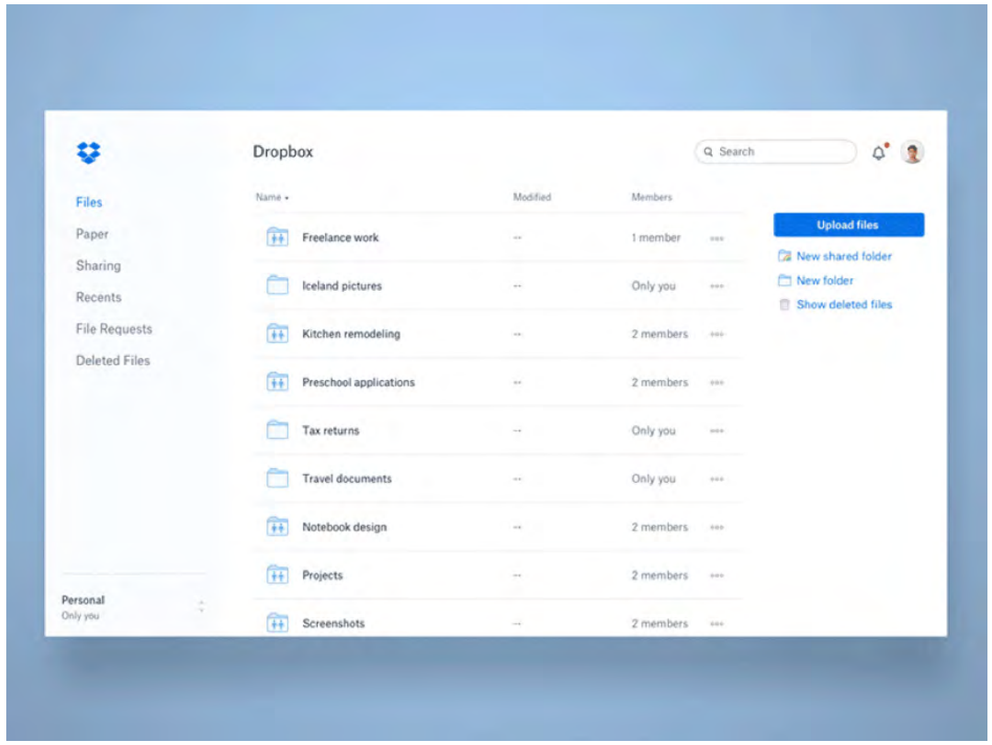
MICROSOFT SHAREPOINT

Microsoft SharePoint facilitates collaboration, document management, and intranet content management.



DROPBOX

Dropbox provides file hosting, sharing, and collaboration features with robust synchronization across devices.



Examples of Common Groupings of DE Tools By Application

Now that you understand what each application does, here are a set of applications that use a combination of digital-engineering tools together in order to solve a problem.

1. APPLICATION: SPACE-BASED OPTICAL SYSTEM ORBITAL ANALYSIS

Problem that this combination solves: Accurately predicts the optical performance of a system through a changing environment as a consequence of varying solar loading throughout an orbit.

Geometry Representations (CAD)	**Mesh Generation (FE generation)**	**Structures (NASTRAN, ANSYS, etc.)**	**Thermal (ANSYS Thermal, Spaceclaim)**	**Optics (Code V, ANSYS Optics Studio, Fred)**
3D models of satellite geometry	Mesh for satellite thermal analysis & thermo-elastic analysis	Distortion effects due to thermal gradients and material in homogeneity and mismatches	Thermal modeling of solar exposure	Lens design for optical systems

2. APPLICATION: FORMULA 1 RACING AERODYNAMICS

Problem that this combination solves: Ensures optimal aerodynamic performance (maximize downforce and minimize drag), safety, and real-time system monitoring for peak race conditions.

Geometry Representations (CAD)	Mesh Generation (FE generation)	CFD (Computational Fluid Dynamics)	Control Systems (MATLAB Simulink)
Car outer mold line, and aerodynamic surfaces (wings, Venturi blades, etc.) design in CAD	Aerodynamic mesh for CFD	Aerodynamic performance testing (OpenFOAM, Fluent)	Modeling Drag Reduction System effects

3. APPLICATION: HIGH END CONSUMER ELECTRONICS FORM FACTOR DESIGN

Problem that this combination solves: Optimizes thermal management and structural integrity, leading to efficient and reliable devices.

Geometry Representations (CAD)	Structures (NASTRAN, ANSYS, etc.)	Thermal (ANSYS Thermal, Spaceclaim)	Electronics (ECAD, EM)	Circuit Board Analysis (ICEPack)
PCB casing design	Vibration and drop analysis	Heat dissipation in compact spaces	EMC compliance	Board cooling and thermal management

4. APPLICATION: CONSUMER ELECTRIC VEHICLE POWER MANAGEMENT

Problem that this combination solves: Enhances energy efficiency, durability, and safety of EVs by integrating thermal, structural, and electrical simulations.

Geometry Representations (CAD)	Mesh Generation (FE generation)	Thermal (ANSYS Thermal, Spaceclaim)	Optics (Code V, ANSYS Optics Studio, Fred)
Laser containment chamber design	Mesh for thermal and structural load	Heat management in containment	Beam shaping for energy dispersion

5. APPLICATION: HYPERSONIC FLIGHT PERFORMANCE

Problem that this combination solves: Simulates extreme flight environments, ensuring safety, performance, and thermal durability at hypersonic speeds.

Geometry Representations (CAD)	Mesh Generation (FE generation)	Thermal (ANSYS Thermal, Spaceclaim)	CFD (Computational Fluid Dynamics)
Hypersonic vehicle geometry	Mesh for aerodynamic simulation	Heat dissipation from air friction	Aero heating and lift to drag assessment (Cart3D, Fluent)

6. APPLICATION: HIGH ENERGY LASER STRAY ENERGY CONTAINMENT

Problem that this combination solves: Integrates thermal, optical, and structural tools to design safe, efficient high-energy laser systems.

Geometry Representations (CAD)	Mesh Generation (FE generation)	Thermal (ANSYS Thermal, Spaceclaim)	Optics (Code V, ANSYS Optics Studio, Fred)
Laser containment chamber design	Mesh for thermal and structural load	Heat management in containment	Beam shaping for energy dispersion

7. APPLICATION: FLIGHT CERTIFICATION, FLUTTER DETECTION

Problem that this combination solves: Verifies and certifies aircraft designs to meet safety and performance standards under high-stress environments.

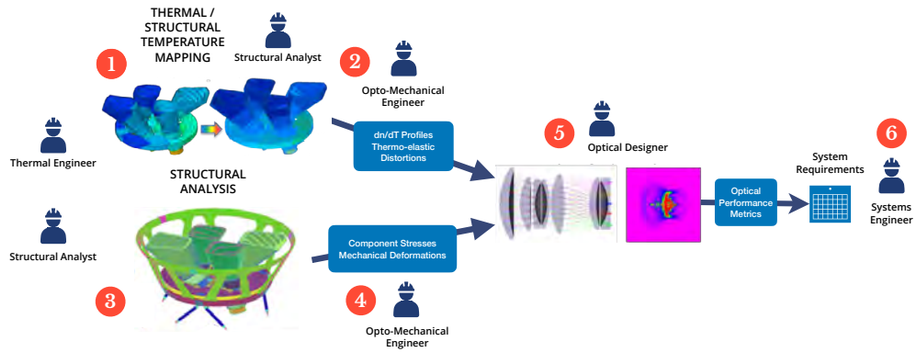
Geometry Representations (CAD)	Mesh Generation (FE generation)	Structures (NASTRAN, ANSYS, etc.)	CFD (Computational Fluid Dynamics)
Aircraft outer mold line and wing stiffness	Mesh for structural and aero loads	Flutter mode detection	Aerodynamic testing of flutter modes (CFD, Fluent)

The Role of Digital Threading and Automation in Connecting DE Tools

DE tools like these can be combined in several ways using digital threads, often with a human in the loop to verify and direct the next set of activities that the system should take. Let's look at a single example of a workflow with varying levels of digital threading and automation to better understand how tools can connect with each other and the value that threading and automation can provide.

EXAMPLE 1: AN OPEN-LOOP SYSTEM WITH NO AUTOMATION

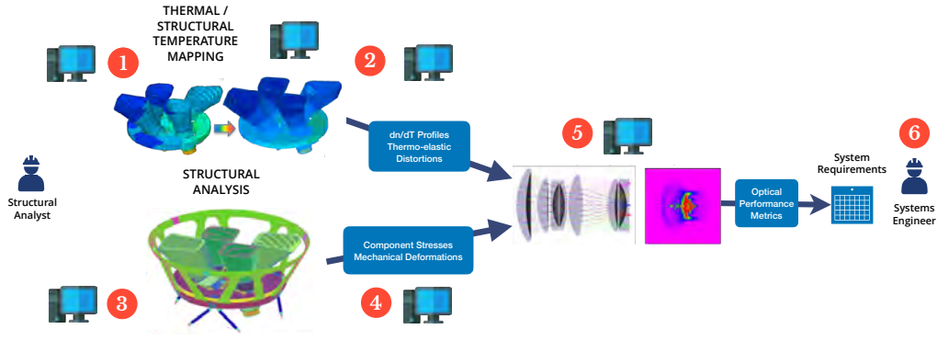
In this example, we'll orient ourselves and see how engineers connect different digital engineering tools by hand, not using a digital thread. For our first example, a team wants to understand how much performance degradation will occur in an optical system used on a satellite. The engineers know that in orbit the temperatures on the device will fluctuate tremendously, and since the device is built out of many different materials, the temperature fluctuations will cause the optics to distort. To understand how their design will perform, they can use the outputs from one simulation to provide inputs to another simulation. Let's walk through this example step by step:



1. We start with existing structural and thermal models that were built by structural and thermal engineers. In this step:
    a. The thermal engineer executes an analysis of a model (a simulation) for the device as if it were to pass through an entire orbit around the earth.
    b. The thermal engineer then hands off the temperature distributions (from the thermal simulation) to the structural engineer, who will spatially map those data on to the structural model.
    c. The structural engineer then creates a simulation that merges the output of both the thermal and structural simulations to understand how the optical system will deform throughout the orbit around the earth.
2. An opto-mechanical engineer now takes the thermal and structural simulation outputs (temperatures and displacements) and maps them to an optical model.
3. Meanwhile, a second structural engineer analyzes other environmental loads that will also affect the system, such as the effect of gravity on the optical system (technically "gravity release" which occurs when a system is built on earth, but then is deployed to work in space).
4. The outputs from the second structural engineer are then mapped to an optical model by another opto-mechanical engineer.
5. An optical designer now combines all the data from the previous simulations to more accurately characterize the performance of the optical system design.
6. The results of this more complete design are given to a systems engineer who validates it against the requirements.

This is a typical workflow for many organizations. Because it lacks any automation, there are potential errors that can be introduced whenever there is a handoff or exchange of data. It also means the people wait for other people to finish their work and verify it before sending it to the next person in the workflow, which increases the time to complete a single iteration of this workflow. Lastly, there's no traceability in this system, so 6 months later it's impossible to understand how the results were determined.

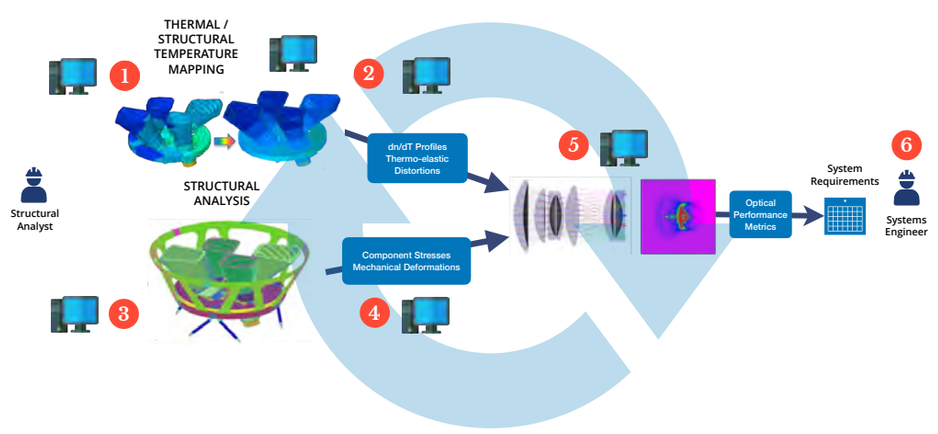
EXAMPLE 2: AN OPEN-LOOP SYSTEM WITH FULLY AUTOMATED WORKFLOW



Here, we see the same system however all the digital engineering tools are connected together by a digital thread – which means that the whole process is now automated. We only need one structural engineer to kick off the process, and automatically all the data flows from tool to tool until it gets to the systems engineer, who evaluates the results against the requirements.

Some advantage is gained here, in that if someone thinks there's, say, a better material to use for some part of the device, a structural engineer can make one quick change in one place, and re-start the whole automated pipeline – an enormous time and cost savings! The problem, however, is that the one engineer who is in charge of this system now has to push a change every time the systems engineer determines that the design isn't meeting the requirements. Starting every iteration is still a manual task, and humans have to guide the evolution of which parameters to change and what values to use.

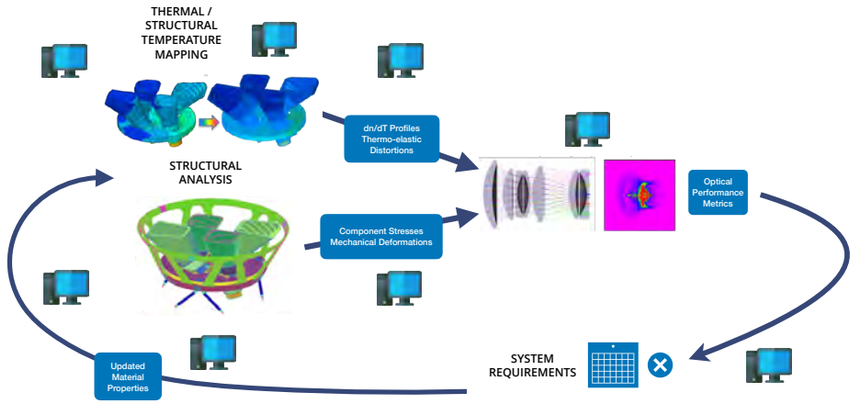
EXAMPLE 3: AN OPEN-LOOP SYSTEM WITH FULLY AUTOMATED WORKFLOW, AND A LITTLE EXTERNAL AUTOMATION



Now if we were to add a little more automation to the workflow, say, a Monte Carlo simulation that could tweak some material properties, we open up the ability for the system to try out numerous variations all by itself. A structural engineer could specify a range of values for the system to test, and then the system can run the entire pipeline over and over, generating a table that provides a variety of outputs to the systems engineer for evaluation.

The same engineer who might have to be at the office all weekend trying a variety of values, can now leave on Friday and know that on Monday morning the systems engineer will have a huge set of results to evaluate. The remaining downside is that there's still a huge set of results that the systems engineer has to manually evaluate against the requirements, which is quite time consuming.

EXAMPLE 4: A CLOSED-LOOP SYSTEM WITH FULLY AUTOMATED WORKFLOW, AND ROBUST EXTERNAL AUTOMATION



Now let's add some intelligent optimization algorithm that checks the results against the requirements document. In this case, the only result that's provided to the systems engineer is the one that actually meets all the requirements. This kind of system can vary a number of values, try different kinds of techniques, and learn from the results – tuning the approaches to find successful answers even faster.

In this situation, automation and digital threading truly shine. Not only do they deliver the ability to audit and trace every step of a design's development, they also free up engineers' time for higher-value tasks and offer a scalable solution capable of adapting to growing project complexity. And most critically, they enable the discovery of innovative solutions to complex problems that go beyond the limits of human intuition.

Building The Auditorium: The Required Technical Infrastructure

Having musicians with great instruments isn't enough – they need to play together to achieve great results and then they need to work in the right place. A jazz trio might be out of place in a rock club, and a symphony orchestra wouldn't work effectively in a New York City jazz lounge.

In the same way we can't provide a single guideline for what tools you might need at different stages of a project, (just like we couldn't have told the Beatles which additional musicians they might need to make a new record), we can suggest a typical path for a certain class of problems.

Obviously small engineering projects like, say, developing a case for sunglasses, will require fewer engineers, fewer digital-engineering tools, and less testing than developing, say, a Formula 1 racecar. They will also require fewer VMs, less storage, and be less costly from an IT perspective. For our example case, let's use the development of an unmanned aerial vehicle for the US military and explore the tools needed and the IT requirements.

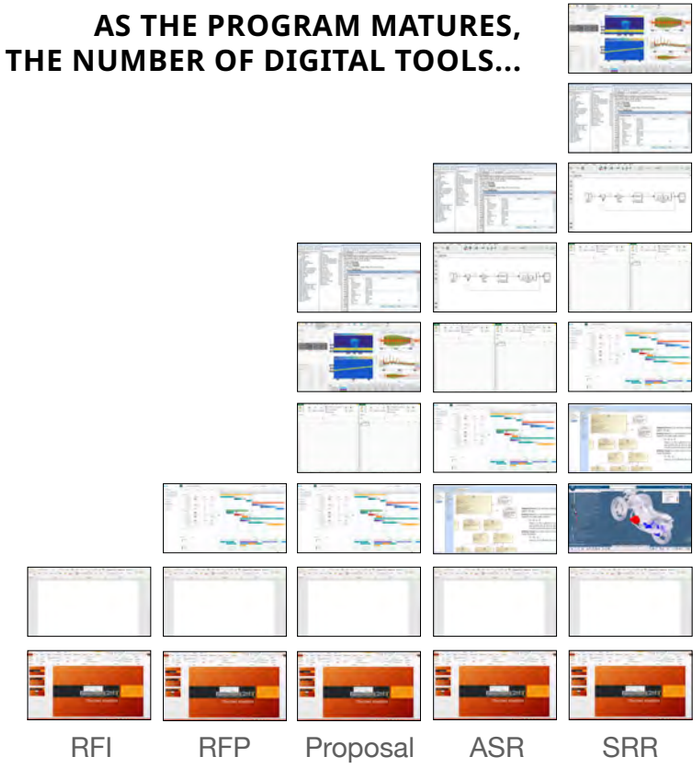
In the next chart, you'll see the stage of development, the tools needed for each stage, and a rationale for what those tools are used for. The more these tools are linked together through digital threads, the faster the development, testing and refinement will be – and it can enable complete digital design and certification before the drone is developed.

Digital Engineering Tools Needed By Stage For An Unmanned Aerial Vehicle

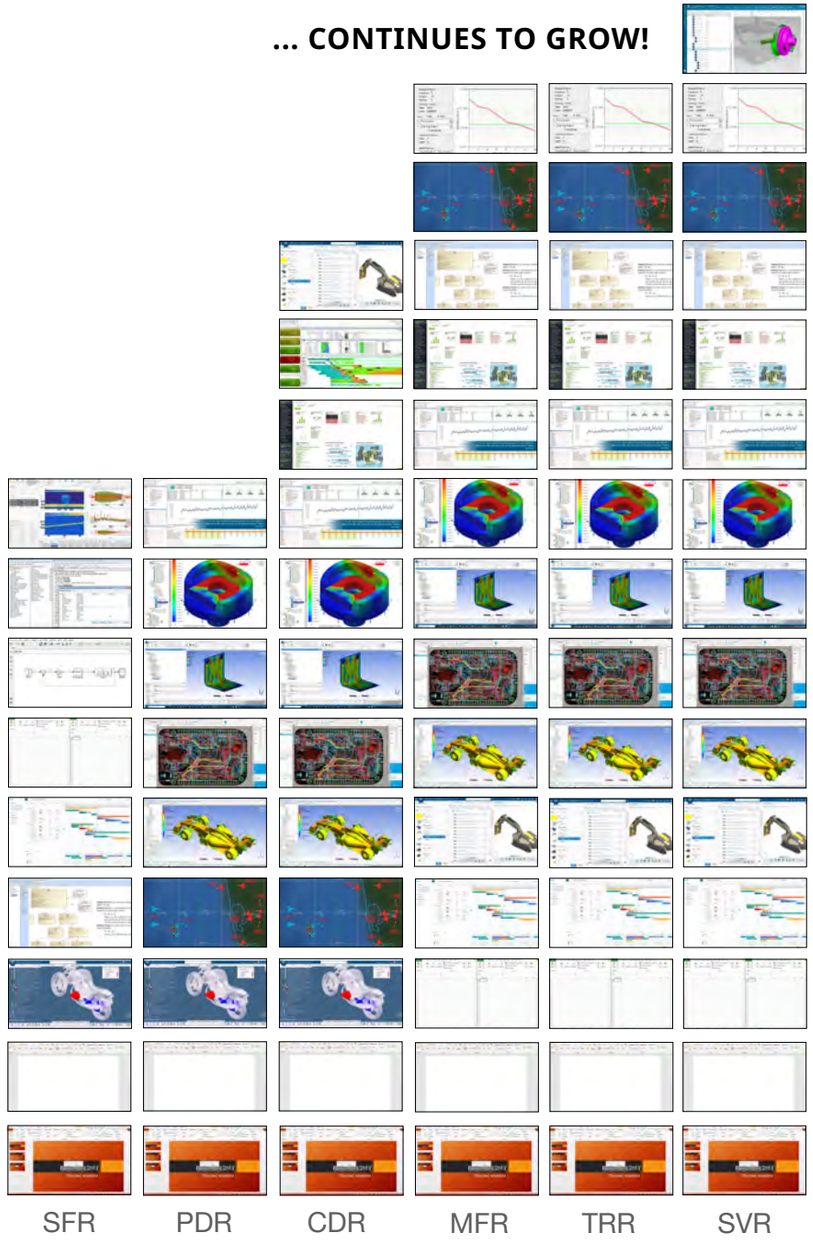
Stage	Tools Needed	Rationale
RFI	Microsoft Word, Microsoft PowerPoint	In the beginning, when you respond to an RFI, it is typically re-using past analysis and making quick graphics in PowerPoint
RFP	MS-Word, MS-PowerPoint,MS-Project	Microsoft Word, PowerPoint, and Project are popular choices for building RFPs because they are widely available, user-friendly, and offer features that streamline proposal creation. Word excels at detailed written content, PowerPoint facilitates visual presentations, and Project aids in project planning and timeline development. These tools together provide a convenient suite for crafting comprehensive RFP responses.
Proposal	MS-Word, MS-PowerPoint,MS-Project, MS-Excel, Cameo (Requirements Tools, Architecture Tools, Low Fidelity Analysis tools)	Proposal development often relies on a suite of Microsoft Office tools like Word, PowerPoint, and Excel, alongside specialized software like Cameo Systems Modeler. Word provides a canvas for detailed descriptions and scope outlines, while PowerPoint allows for compelling visual presentations of key ideas and timelines. Excel enables data analysis and cost modeling, and Project facilitates detailed project planning. Cameo Systems Modeler aids in capturing requirements, designing system architectures, and performing preliminary analysis. This combination of tools empowers proposal writers to craft comprehensive, well-structured, and persuasive responses.
ASR	Matlab, Cameo, MS-Project, MS-Excel	During the ASR stage, we use Matlab for advanced modeling and simulation to evaluate performance trade-offs, Cameo for capturing and analyzing system architectures and requirements in a model-based systems engineering (MBSE) framework, MS-Project for managing timelines and schedules effectively, and MS-Excel for conducting detailed cost analyses, organizing data, and performing sensitivity analyses to support trade-space decisions
SRR	Matlab, Cameo, Siemens Teamcenter, MS-Project, MS-Excel	During the SRR stage, we use Matlab for analyzing and validating system performance against requirements, Cameo to model and assess system requirements within a structured MBSE framework, Siemens Teamcenter for managing configuration and ensuring traceability of requirements, MS-Project for tracking progress and aligning schedules with development milestones, and MS-Excel for organizing data, performing requirements gap analysis, and generating reports to support decision-making
SFR	Matlab, Cameo, Siemens Teamcenter, MS-Project, MS-Excel	During the SFR stage, we use Matlab to simulate and validate system functions and interactions, Cameo to ensure consistency and completeness in the functional architecture within an MBSE framework, Siemens Teamcenter for managing functional baselines and maintaining traceability between functions and requirements, MS-Project to track progress and ensure alignment with the project schedule, and MS-Excel for detailed data analysis, functional gap identification, and reporting to support functional validation and decision-making
PDR	Ansys HFSS, Autodesk Eagle, Siemens Star CCM+, Ansys Mechanical, AFSIM Mission Effects Modeler, Siemens NX, Dassault Delmia	During the PDR stage, we use Ansys HFSS for high-frequency electromagnetic simulations to validate antenna and RF designs, Autodesk Eagle for detailed PCB design and layout, Siemens Star CCM+ for advanced computational fluid dynamics analysis, Ansys Mechanical to simulate structural integrity under various loads, AFSIM Mission Effects Modeler to assess mission-level performance and operational impacts, Siemens NX for 3D mechanical design and digital twin development, and Dassault Delmia for simulating manufacturing processes to ensure design feasibility and production readiness
CDR	Ansys HFSS, Autodesk Eagle, Siemens Star CCM+, Ansys Mechanical, AFSIM Mission Effects Modeler, Siemens NX, Dassault Delmia, Dassault Delmia Quintic, Patsnap IP modeler	During the CDR stage, we use Ansys HFSS for finalizing electromagnetic simulations of high-frequency components, Autodesk Eagle for verifying PCB designs and layouts, Siemens Star CCM+ for advanced fluid dynamics simulations to ensure thermal and aerodynamic performance, and Ansys Mechanical to confirm structural integrity under operational conditions. AFSIM Mission Effects Modeler is employed to validate mission-level performance against design specifications, Siemens NX for refining detailed mechanical designs and integrating components, and Dassault Delmia for ensuring manufacturability through process simulation. Additionally, Dassault Delmia Quintic is used for optimizing supply chain and logistics considerations, while Patsnap IP Modeler supports intellectual property assessments to confirm innovation alignment with design outcomes
MFR	Matlab, Ansys HFSS, Autodesk Eagle, Siemens Star CCM+, Ansys Mechanical, AFSIM Mission Effects Modeler, Siemens NX, Dassault Delmia, Dassault Delmia, Cameo, Siemens Teamcenter, MS-Project, MS-Excel	During the MFR stage, we use Matlab for analyzing and optimizing system performance metrics critical to manufacturing feasibility, Ansys HFSS for validating the manufacturability of RF and electromagnetic designs, and Autodesk Eagle for ensuring PCB layouts meet production standards. Siemens Star CCM+ is employed for fluid dynamics simulations to validate manufacturing tolerances, while Ansys Mechanical ensures structural components are ready for production with necessary durability. AFSIM Mission Effects Modeler helps confirm operational effectiveness of the manufactured system, Siemens NX supports the refinement of mechanical designs for production, and Dassault Delmia simulates manufacturing workflows to optimize efficiency. Cameo ensures model-based traceability and consistency between system designs and manufacturing processes, Siemens Teamcenter manages configuration and integrates manufacturing data, MS-Project tracks production schedules and readiness, and MS-Excel is used for cost analysis, production data organization, and detailed reporting to ensure manufacturing goals are met
TRR	Matlab, Ansys HFSS, Autodesk Eagle, Siemens Star CCM+, Ansys Mechanical, AFSIM Mission Effects Modeler, Siemens NX, Dassault Delmia, Dassault Delmia, Cameo, Siemens Teamcenter, MS-Project, MS-Excel	During the TRR stage, we use Matlab for test data simulation, analysis, and validation of test scenarios, Ansys HFSS for verifying electromagnetic and RF performance in test environments, and Autodesk Eagle to confirm PCB designs meet testing specifications. Siemens Star CCM+ is utilized for fluid and thermal analyses to validate test conditions, while Ansys Mechanical ensures structural components are prepared for stress and load testing. AFSIM Mission Effects Modeler helps evaluate system performance under mission-level test scenarios, Siemens NX is used for refining designs based on test setup requirements, and Dassault Delmia supports the simulation of test workflows and procedures. Cameo provides traceability to ensure alignment of tests with system requirements, Siemens Teamcenter manages test configuration and documentation, MS-Project tracks test schedules and milestones, and MS-Excel is used for organizing test data, analyzing results, and generating reports to support test readiness decisions
SVR	Matlab, Ansys HFSS, Autodesk Eagle, Siemens Star CCM+, Ansys Mechanical, AFSIM Mission Effects Modeler, Siemens NX, Dassault Delmia, Dassault Delmia, Cameo, Siemens Teamcenter, MS-Project, MS-Excel	During the SVR stage, we use Matlab for analyzing and validating test data against system requirements, Ansys HFSS to confirm electromagnetic and RF performance, and Autodesk Eagle to ensure PCB designs meet verification standards. Siemens Star CCM+ supports fluid and thermal performance assessments under test conditions, while Ansys Mechanical validates structural integrity during system testing. AFSIM Mission Effects Modeler helps evaluate mission-level system performance in verification scenarios, Siemens NX is used for refining and validating physical designs, and Dassault Delmia ensures manufacturability insights are consistent with verification results. Cameo provides traceability to confirm all system requirements are satisfied during testing, Siemens Teamcenter manages test data, configurations, and documentation, MS-Project tracks progress and ensures alignment with verification schedules, and MS-Excel organizes test results, enables data analysis, and generates reports to support verification decisions

Think about it like you're building an auditorium for your musicians: a jazz trio is very comfortable in a small intimate lounge with a few lights, a rock band needs a larger club with sound amplification, and a symphony orchestra needs a concert hall designed with even lighting, and acoustic design to create perfect listening conditions when it's filled with people.

AS THE PROGRAM MATURES, THE NUMBER OF DIGITAL TOOLS...



... CONTINUES TO GROW!



Technical Requirements Needed By Stage For An Unmanned Aerial Vehicle

We can take the Unmanned Aerial Vehicle example a step further, by accounting for the storage type (e.g., shared drive vs. scalable cloud), computation type (e.g., local vs. cloud), and—in those cases where cloud compute is utilized—required performance (i.e., the RAM, number of cores, and hours of compute time). These storage and compute decisions are highly tailorable based on your preferences and the specific needs of the project.

Let's go one step deeper with a table that shows the details about the tools, storage and compute type as well as performance.

HERE ARE SOME DEFINITIONS AND CONSIDERATIONS RELATED TO PERFORMANCE:

- **RAM (Random Access Memory):** The computer's short-term memory that temporarily stores data and instructions for quick access by the CPU during processing. In simulations, RAM stores the models and datasets being processed, directly impacting the ability to handle large or complex simulations efficiently.
  - » **Standard Memory Nodes (e.g., 2 GB RAM per core):** Suitable for general-purpose computing and simulations with moderate memory requirements. These are applied when simulations involve smaller datasets or less complex models that do not require extensive memory resources.
  - » **High-Memory Nodes (e.g., 4 GB RAM per core):** Ideal for simulations needing more memory per core due to larger datasets or higher complexity. They are commonly used in high-fidelity Computational Fluid Dynamics (CFD) and Finite Element Analysis (FEA) simulations where finer mesh resolutions and detailed models demand additional memory. We assume High-Memory Nodes in our example below.
  - » **Ultra-Memory Nodes (e.g., 8 GB RAM per core or more):** Necessary for extremely large or highly detailed simulations that require significant memory capacity. These are applied in cases such as large-scale fluid dynamics, detailed electromagnetic simulations, or when dealing with massive datasets that cannot be efficiently processed on nodes with less memory.
- **Cores:** Independent processing units within a computer's CPU that execute program instructions simultaneously, enabling parallel computation. In simulations, multiple cores allow different parts of the simulation to be computed at the same time, significantly reducing the total computation time for resource-intensive tasks.
- **Core-Hours:** A unit of computational resource usage representing one CPU core operating for one hour, used to measure and bill processing time in computing tasks. In simulations, core-hours quantify the total computational effort required, helping estimate the computational cost and time needed to complete a simulation. The cost per Core-Hour is impacted by the RAM for each Core.

Stage	Tool and Purpose	Storage Type	Compute Type - Performance (if applicable)
RFI	Word for RFI	Shared Drive	Local
	PowerPoint for RFI Deck	Shared Drive	Local
RFP	Word for RFP	Shared Drive	Local
	PowerPoint for RFP Deck	Shared Drive	Local
	MS Project for RFP WBS	Shared Drive	Local
Proposal	Word for Proposal	Shared Drive	Local
	PowerPoint for Proposal Deck	Shared Drive	Local
	MS Project for Proposal WBS	Shared Drive	Local
	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	MATLAB for Low-Fidelity Analytical Models	Shared Drive	Local
	MATLAB for Tradespace Model	Shared Drive	Local
ASR	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	MS Project for Gantt Chart Model	Shared Drive	Local
	Excel for Cost Model	Shared Drive	Local
SRR	MATLAB for Low-Fidelity Analytical Models	Shared Drive	Local
	MATLAB for Tradespace Model	Shared Drive	Local
	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	3DX for Product Lifecycle Manager	Collaboration Database	Local
	MS Project for Gantt Chart Model	Shared Drive	Local
	Excel for Cost Model	Shared Drive	Local
SFR	MATLAB for Low-Fidelity Analytical Models	Shared Drive	Local
	MATLAB for Tradespace Model	Shared Drive	Local
	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	3DX for Product Lifecycle Manager	Collaboration Database	Local
	MS Project for Gantt Chart Model	Shared Drive	Local
	Excel for Cost Model	Shared Drive	Local
PDR	Ansys HFSS for Electromagnetic Frequency Model	Scalable Cloud	HPC - 1280 core hours
	Autodesk Eagle for Electronic System Model	Shared Drive	Local
	Ansys Fluent for Computational Fluid Dynamics (CFD)	Scalable Cloud	HPC - 3840 core hours
	NASTRAN for Finite Element Analysis	Scalable Cloud	HPC -1920 core hours
	AFSIM for Mission Effects Model	Shared Drive	Local
	3DX for CAD Models	Collaboration Database	Local
	Dassault Delmia for Manufacturing Model	Collaboration Database	Local
CDR	Ansys HFSS for Electromagnetic Frequency Model	Scalable Cloud	HPC - 5120 core hours
	Autodesk Eagle for Electronic System Model	Shared Drive	Local
	Ansys Fluent for Computational Fluid Dynamics (CFD)	Scalable Cloud	HPC - 15360 core hours
	NASTRAN for Finite Element Analysis	Scalable Cloud	HPC - 7680 core hours
	AFSIM for Mission Effects Model	Shared Drive	Local
	3DX for CAD Models	Collaboration Database	Local
	Dassault Delmia for Manufacturing Model	Collaboration Database	Local
	Dassault Delmia Quintiq for Supply Chain Model	Collaboration Database	Local
	Patsnap for IP Model	Shared Drive	Local

MFR	Ansys Fluent for Computational Fluid Dynamics (CFD)	Scalable Cloud	HPC - 960 core hours
	NASTRAN for Finite Element Analysis	Scalable Cloud	HPC - 480 core hours
	Ansys HFSS for Electromagnetic Frequency Model	Scalable Cloud	HPC - 320 core hours
	Autodesk Eagle for Electronic System Model	Shared Drive	Local
	MATLAB for Low-Fidelity Analytical Models	Shared Drive	Local
	MATLAB for Tradespace Model	Shared Drive	Local
	AFSIM for Mission Effects Model	Shared Drive	Local
	3DX for CAD Models	Collaboration Database	Local
	Dassault Delmia for Manufacturing Model	Collaboration Database	Local
	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	3DX for Product Lifecycle Manager	Collaboration Database	Local
	MS Project for Gantt Chart Model	Shared Drive	Local
	Excel for Cost Model	Shared Drive	Local
TRR	Ansys Fluent for Computational Fluid Dynamics (CFD)	Scalable Cloud	HPC - 960 core hours
	NASTRAN for Finite Element Analysis	Scalable Cloud	HPC - 480 core hours
	Ansys HFSS for Electromagnetic Frequency Model	Scalable Cloud	HPC - 320 core hours
	Autodesk Eagle for Electronic System Model	Shared Drive	Local
	MATLAB for Low-Fidelity Analytical Models	Shared Drive	Local
	MATLAB for Tradespace Model	Shared Drive	Local
	AFSIM for Mission Effects Model	Shared Drive	Local
	3DX for CAD Models	Collaboration Database	Local
	Dassault Delmia for Manufacturing Model	Collaboration Database	Local
	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	3DX for Product Lifecycle Manager	Collaboration Database	Local
	MS Project for Gantt Chart Model	Shared Drive	Local
	Excel for Cost Model	Shared Drive	Local
SVR	Ansys Fluent for Computational Fluid Dynamics (CFD)	Scalable Cloud	HPC - 160 core hours
	NASTRAN for Finite Element Analysis	Scalable Cloud	HPC - 80 core hours
	Ansys HFSS for Electromagnetic Frequency Model	Scalable Cloud	HPC - 60 core hours
	Autodesk Eagle for Electronic System Model	Shared Drive	Local
	MATLAB for Low-Fidelity Analytical Models	Shared Drive	Local
	MATLAB for Tradespace Model	Shared Drive	Local
	AFSIM for Mission Effects Model	Shared Drive	Local
	3DX for CAD Models	Collaboration Database	Local
	Dassault Delmia for Manufacturing Model	Collaboration Database	Local
	Cameo for Requirements	Collaboration Database	Local
	Cameo for Operational Architecture	Collaboration Database	Local
	Cameo for Functional Architecture	Collaboration Database	Local
	Cameo for System Architecture	Collaboration Database	Local
	3DX for Product Lifecycle Manager	Collaboration Database	Local
	MS Project for Gantt Chart Model	Shared Drive	Local
	Excel for Cost Model	Shared Drive	Local

4

From Virtuoso To Maestro: The Engineer As Composer, Conductor, Producer

Not long ago, we seldom made a hard distinction between a mechanical engineer, a human-factors engineer, and an industrial designer. In fact, they would often overlap in skill sets and knowledge. In 1941, mechanical engineer Carl Kronmiller and industrial designer Henry Dreyfuss created the now-famous round Honeywell thermostat. Dreyfuss didn't have a formal education as an engineer or as an industrial designer—he was an apprentice to industrial designer Norman Bel Geddes and, during his apprenticeship, produced more than 250 stage sets for theaters before founding his own firm, Henry Dreyfuss Associates.

But Dreyfuss was focused on the intersection of engineering and humans and famously said, "when the point of contact between the product and people becomes a point of friction, then the industrial designer has failed." He went on to design the "Princess" phone for Bell Laboratories, tailored to fit comfortably in the hand of a teenage girl, a refrigerator for General Electric, alarm clocks for Westclox, vacuum cleaners for Hoover, the J-3 Hudson locomotive, tractors for John Deere, and more.

In addition to Dreyfuss, many other influential figures contributed to shaping the future we live in now: Raymond Loewy, who designed the Coca-Cola vending machine and streamlined trains and cars; Dieter Rams, who produced groundbreaking designs for Braun, including calculators, speakers, and clocks; Thomas Edison, whose prolific list of inventions revolutionized technology; and Thomas Watson Jr., who transformed IBM into a leader in computing. Despite a global population larger than ever, the number of individuals who hold this exalted position of innovator and inventor has noticeably decreased. Outside of rare figures like Steve Jobs or Elon Musk, we seldom hear of innovators who can oversee and deeply contribute to every aspect of a complex project. Why?

3 Factors Contributing To The Shift In The Nature Of Innovation

Increased Complexity of Modern Systems: Modern products, especially in technology, are comprised of vast networks of interdependent systems. These systems are so intricate that they require teams of specialized experts to manage. It is increasingly rare for one person to have a comprehensive understanding of all elements, from the user interface to the technical inner workings.

Team-Based Approaches: Today's design and development landscape relies heavily on collaborative work. Large teams of specialists contribute to projects with segmented expertise, making it less common for a singular visionary to oversee the entire process from start to finish. Software development, artificial intelligence, and integrated tech products all demand multifaceted approaches that exceed the capacity of any single individual.

Corporate Scale and Global Operations: Companies today operate on a scale that is vast and spread across multiple departments and international locations. Managing this level of complexity requires decentralized control, further limiting the ability for one person to maintain comprehensive oversight. In the past, a singular expert could lead projects like Dreyfuss did, imagining the full system—from a thermostat's placement and user interaction to its internal mechanisms and how they would respond to environmental changes.

For the kind of engineering we do now, it's not practical for one person to deeply understand every aspect of a project. Take, for example, the space station: does any single individual know how every element of each system and subsystem works in detail? Our ability to create things that were in the realm of science fiction just 50 years ago stems from our capacity to build increasingly complex systems that maintain robustness and reliability.

The New Frontier of Visionary Engineering

Can we capture the essence of these famous innovators—those who could orchestrate every part of a project—by leveraging modern technologies in new ways? Can we push the boundaries of engineering and development to create even more powerful solutions than those once dreamt up in science fiction?

Let's return to the world of music once more. Consider Quincy Jones, who produced Michael Jackson's Thriller, George Martin of Beatles fame, or Prince. Each of these figures could compose, arrange, play, and produce music that was not only commercially successful but groundbreaking. What they shared was an exceptional ability to rapidly develop and refine ideas, navigate technical and creative constraints, and scale their results to reach massive audiences.

These trailblazers remind us that while today's challenges may be more complex and dispersed, the ability to blend creativity, engineering, and vision will always be crucial. The next wave of breakthroughs requires a new approach and a new kind of engineer.

A New Kind Of Engineer

The development of a new kind of engineer requires that we begin to teach our current engineers, and the next generation of engineers, to think holistically and to understand the interconnectedness of all engineering disciplines, as well as some non-engineering disciplines.

The 3 Types Of Engineer:

THE SOLOIST

In some organizations we have very talented solo engineers, much like a violinist who plays in an orchestra but isn't leading the orchestra or the violin section. This person knows their discipline with an extreme amount of depth, and can use their tools extremely effectively to accomplish a result. In an orchestra, the vast majority of the players are in this group – and their value is immense, and will be in the future.

This type of solo digital-engineer will need to learn how adjacent engineering disciplines will use the output of their work. Much like how a violinist knows how to play well with a pianist, the solo engineer will have to ensure that elements of their models work well with other engineering disciplines. For example, this person might know a lot about mechanical engineering and a popular CAD program may be their primary tool. They will learn how to correctly develop their models in ways that enable other people and other tools to leverage their work easily.

THE SECTION LEADER

In an orchestra, the lead violinist sets the tone of the section, helps ensure consistency across the section, interprets the conductor's vision and creates a uniform sound. In our digital-engineering orchestra, we might call that person an engineering section leader.

This engineering leader will have a lot of knowledge of their own discipline as well as some depth in a number of adjacent disciplines and be able to work to help individual players (the solo engineer) to be able to structure their work to make it accessible to both humans and computers, and as a result this person will have to be familiar with other popular software products in adjacent disciplines. For example, this person might be a mechanical engineer by trade, and may have spent significant time working with PLM systems and analysis tools for things like FEA.

They'll also understand how to help others establish best-practices around using their own tools, developing their models so that people who are outside of their team and discipline are able to understand how to interpret the structure and the output of the models. They'll be able to correctly form a system in which several tools can be threaded together in ways that are both easy to use and powerful for the business.

This engineer will also have a solid understanding of software development best practices and use those skills to help create the software connections between models that are useful for both the engineers as well as satisfying the business needs. Using these skills they'll help create semi-closed-loop systems or fully-closed-loop systems so that models can be analyzed and updated to meet requirements, either by bringing in humans to help make judgement calls or by having the set of tools work together to iterate until the problem is solved.

THE MAESTRO

Of course, when we have large complex systems, we need someone to coordinate the section leaders, and this is where we get our orchestra conductor. Our maestro.

Our maestro will have the broadest understanding of the digital engineering orchestra, and they'll work to not only help instill a set of working principles as to how the engineers will work together but also how the sets of engineering section-leads will work together. They will be responsible for connecting a potentially wide range of technologies and engineering (as well as non-engineering) disciplines and tools. They will be able to connect the pods of CAD-PLM-FEA tools, along with project management and communication tools from a software-engineering perspective.

This person will have a deep understanding of systems engineering and enterprise-level integration approaches, ensuring that the outputs from various engineering disciplines contribute cohesively to the broader goals of the organization. They will foster collaboration between engineering teams, ensuring that data, tools, and workflows are aligned to deliver value efficiently and effectively.

The maestro will establish and enforce governance processes that standardize how tools are used, how data is exchanged, and how results are validated. They will drive innovation, identifying emerging technologies and practices that can enhance the overall digital engineering ecosystem.

In addition to technical expertise, this person will possess strong leadership and communication skills to inspire and guide the teams, aligning diverse efforts toward a common vision. They'll ensure that the orchestra's efforts resonate with the strategic priorities of the business, delivering results that are not only technically sound but also aligned with market and organizational needs.

Ultimately, the maestro will be the key to ensuring that the entire digital engineering system operates as a harmonious whole, seamlessly integrating people, tools, and processes to achieve outcomes that no single engineer or team could accomplish alone.

Many engineers of the future will need to learn about:

PHYSICAL SYSTEM REPRESENTATION, AND THE RELATED TOOLS OF:

- Computer Aided Design
- Electronic Computer-Aided Design (ECAD)
- Product Lifecycle Management (PLM)

SIMULATIONS, AND HOW TO EFFECTIVELY USE AND UNDERSTAND THE RESULTS FROM THEM, INCLUDING:

- Low Fidelity Simulations
- Computational Fluid Dynamics
- Finite Element Analysis (FEA)
- Electromagnetic (EM) Simulation Tools
- Dynamic Simulations
- Simulation Managers

Some engineers will also need to learn about how to use Mission Models:

Model Based System-Engineering and System Architecture and Design Tools will be critical for any complex system and will require an understanding of the effective use of Requirements Tools and System Architecture Tools

Computer Science and Software Engineering will be table stakes for all new engineers, and they will have to be able to understand the principles and uses of Integrated Development Environments, Version Control Systems, CI/CD and Software Testing Tools

And last but not least, the productivity and communication tools will always remain a part of the engineering lifecycle. Knowing the answer is useless if you don't know how to communicate it in a way that can affect change. These skills go hand in hand with leadership skills that are necessary to develop any system with a group of other people.

You might be thinking: it would be too hard for any one engineer to be an expert at all these tools and disciplines. And you'd be right. But they don't need to be an expert at all of them – they just need to understand how all the tools work together as a system when connected through robust digital threads. Once they understand how each tool interacts with the others, and what the outputs mean, they'll be able to form new connections that will enable rapid iterations, and instead of writing every note by hand, it will be like having thousands of engineers working at the speed of light.

Modeling For People And Machines: Techniques Of The Digital-Engineer

In the evolution of a musical composition, a composer usually starts by playing a tune they like and finding a quick way to capture the fragile idea. Mozart would sketch his ideas sitting at a table using a fountain pen, and we can see that in his manuscripts he used short strokes that look like they were made at incredible speed and only are designed to be understood by Mozart himself. These sketches were made without notation for the dynamics indicating how to play the notes, or even the harmonies or an underlying bass line. After he worked on an idea, he would transcribe it into standard notation for others to be able to read and understand and when the work was near complete, he would employ copyists who were skilled in reading and transcribing music, to create the individual parts for each musician, a process still done by hand. For works that were meant to be widely distributed, the parts would be engraved and printed by professional music printers and publishers, including Artaria & Co., which worked extensively with Mozart. Artiaria started in 1770 and only closed its doors in 1932!

Our digital engineer, much like Mozart, will work on 3 levels of fidelity. The first level is working alone to sketch out an idea, the second is making the work easier for others to understand, and the third is enabling other computers to use the work. Let's examine the 3 levels of how we might construct a model.

EXAMPLE OF THE 3 LEVELS OF MODELING: FOR ME, FOR OTHERS, FOR MACHINES

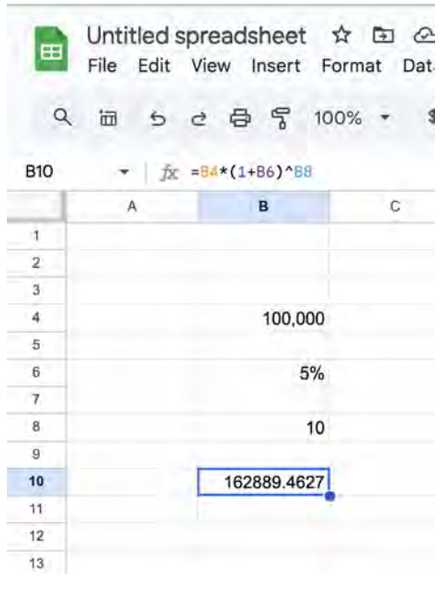
LEVEL 1 - MODELING FOR ME, EXAMPLE OF A GOOGLE SPREADSHEET

When we work alone, most of us tend not to create something in a formal way from the start. Consider using Google Sheets, their collaborative spreadsheet program. If I wanted to understand how much I might be able to make from an investment, I could open up Google Sheets, put in a starting amount, an annual rate of return, a period of time, and run a calculation.

When I make a model (in this case a spreadsheet) for just myself to view, here are the steps that might happen:

- Create simple spreadsheets to manage personal tasks, calculations, or data tracking
- Data is entered manually without structure or organization
- Limited or no use of formulas beyond basic arithmetic (e.g., SUM, AVERAGE)
- No clear formatting or consistent layout, making the spreadsheet difficult to interpret
- No use of advanced features like tables, charts, or conditional formatting
- Focus is solely on solving individual problems or answering immediate questions

Example - Simple spreadsheet with some organization to accomplish a simple requirement

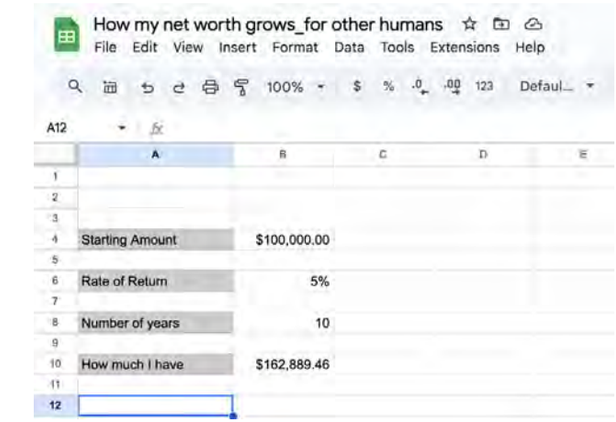


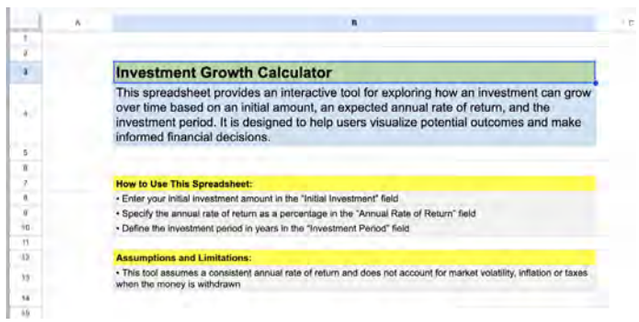
If I read this I might be happy to see that a $100,000 investment with an annual rate of return of 5% over 10 years would yield $163K. Well, technically $162,899 (...and 46 cents, when I round down). This is fine for my purposes, but it's not something I could give to another person if the objective was for us to work together effectively and build out the model. In fact, if I wanted to share this document with someone else it would require that I at least name the file before sharing it. Someone might be able to interpret these data from context, but it looks sloppy and it's very hard to read an amount when it's written "162899.4627", omitting the dollar sign and using four decimals instead of the common two, for pennies.

LEVEL 2 - MODELING FOR OTHERS

Now, if I want someone else to understand this spreadsheet, I would want to add a title, label the various elements, and use standard notation for the dollar amounts. These changes make it far easier for someone to understand this spreadsheet and start to modify and leverage it. I might even include a description about the sheet on a separate tab if I think I'll distribute it to other people who lacked context, like this:

Example - Spreadsheet now with standard notation to better collaborate with others and text to explain context





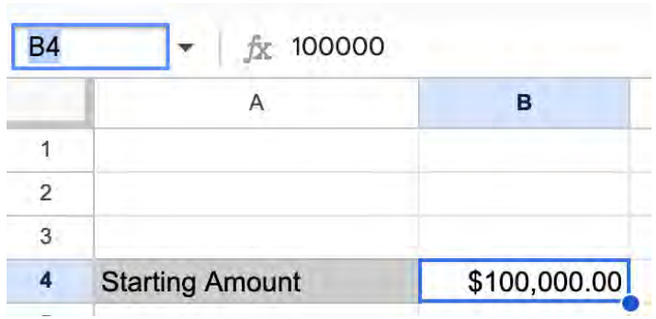
When we develop a model for others, we want to add structures that help other people understand our thinking. Also, this process will help us understand our own thinking, since it requires us to be explicit with our approach. When creating models for someone else, here are activities that might occur:

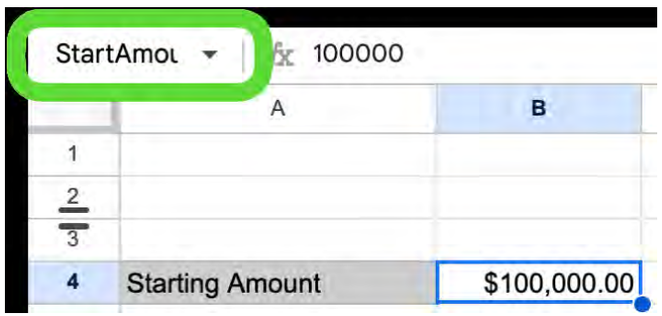
- Create well-organized spreadsheets with structured tables, clear labels, and consistent formatting
- Use intermediate functions (e.g., VLOOKUP, HLOOKUP, IF, INDEX-MATCH) to enhance data analysis and usability
- Incorporate charts and graphs to visualize data for better communication and decision-making
- Apply conditional formatting to highlight key trends, anomalies, or thresholds
- Includes basic documentation (e.g., comments or a “Read Me” sheet) to explain the spreadsheet’s purpose and functionality
- Employs named ranges and data validation to ensure consistency and accuracy
- Share spreadsheets with others with a specific collaboration setting (e.g., “commentor”, “editor”)
- Spreadsheets are designed to handle small-scale data updates without breaking functionality

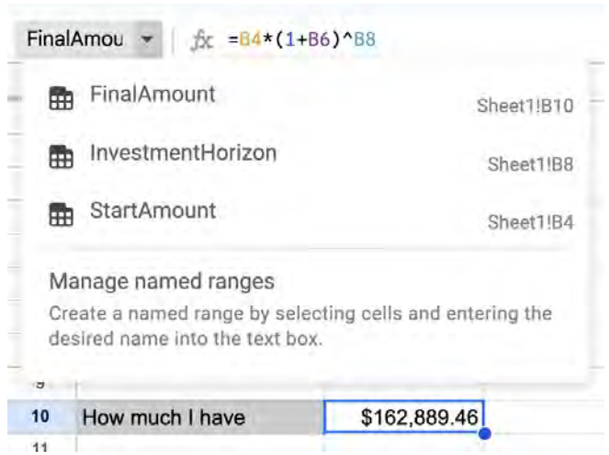
LEVEL 3 - MODELING FOR MACHINES

Now if we wanted to have a computer make sense of the spreadsheet we could hope that a great piece of AI technology could understand it the way a human would, but in reality we can’t count on that occurring without errors and it simply won’t work for extremely complex systems that are unlike other systems in a training set. To enable other computers to understand how to use this sheet, we can create meta-data labels for the variables in individual cells, changing them from “B4” to something like “StartAmount”.

Example - Moving to more advanced spreadsheet to have advanced features like named cells and functions for analysis and automation







This process enables easier data integration when you're exporting or connecting a spreadsheet to another system (e.g., through APIs, scripts, or external tools). The named ranges act as labels for your data which simplifies integration because external systems can query or reference named ranges rather than hardcoded cell addresses. It also makes it easier to maintain, because it becomes easier to update spreadsheets without breaking dependencies. For example, if you move a named cell to a different location, references to it will still work, unlike absolute references (A1), which might need manual adjustments, and it's easier for another person to be able to understand how the sheet is constructed – it's just like commenting software code.

When you create a spreadsheet for other computers to use you may:

- Develop advanced spreadsheets with dynamic functionality, suitable for automation and large-scale data management
- Use complex formulas (e.g., ARRAYFORMULA, XLOOKUP, SUMPRODUCT, nested IF statements) and advanced features like pivot tables for robust analysis
- Implement macros or VBA scripts to automate repetitive tasks, such as data cleaning, calculations, or reporting
- Link spreadsheets to external data sources (e.g., databases, APIs, or live feeds) for real-time updates
- Integrate the spreadsheet with other tools (e.g., Power BI, MS Access, or ERP systems) to create a cohesive workflow
- Build reusable templates with protected cells and dynamic inputs to maintain integrity and scalability
- Design spreadsheets to be machine-readable (e.g., structured data formats like CSV or XML) for seamless integration into larger systems
- Use advanced data validation, error-handling, and input restrictions to minimize user errors
- Fully integrate spreadsheets into the digital thread, enabling data to flow between different systems and tools for enhanced productivity and automation

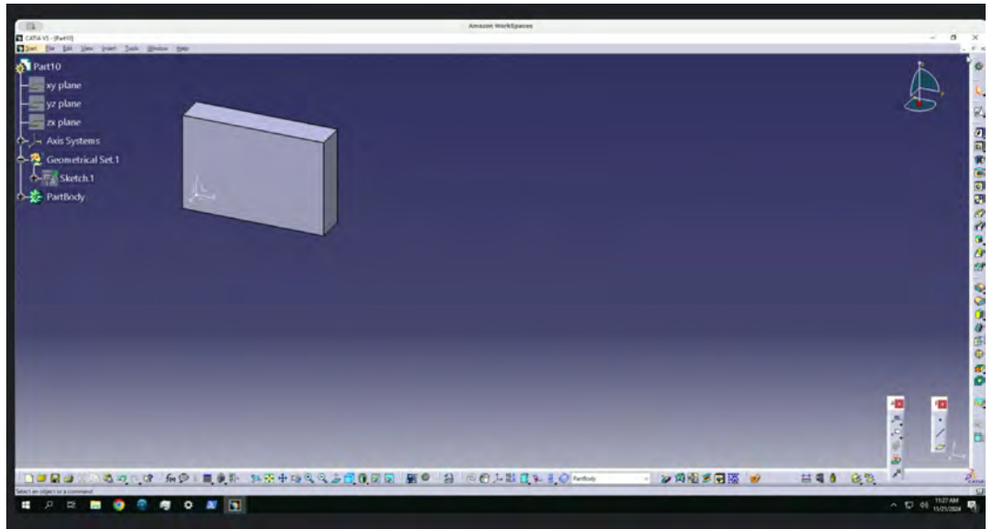
Let's explore examples for how an engineer might create models for each of the 3 levels, by tool type.

CAD - CATIA EXAMPLE

LEVEL 1 - FOR ME

- CAD model and assembly
- Not organized
- No labeling
- No linkages to other models
- Has errors when dimension need medium to large changes

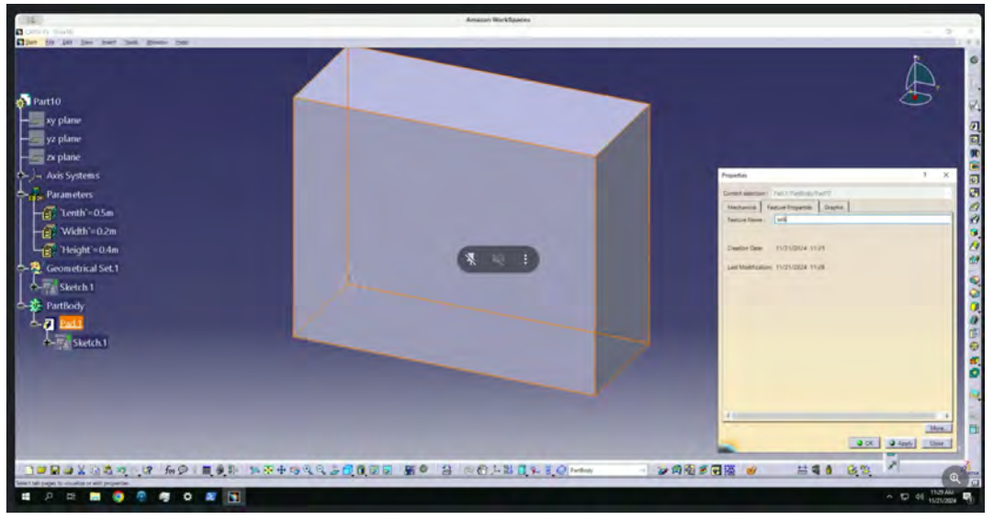
Example Level 1 - Simple CAD model, lack of labeling, no thought of integration with other models or systems



LEVEL 2 - FOR OTHER HUMANS

- Well organized CAD model
- Follows most corporate best practices of the organization and best practices for the particular DE Tool
- Organized tree structure
- Clear labeling
- Some user defined parameters
- Linking from other dependant models
- Utilizes Best Practices and data reuse techniques
- Material assignment and relevant material properties defined in the material library for use

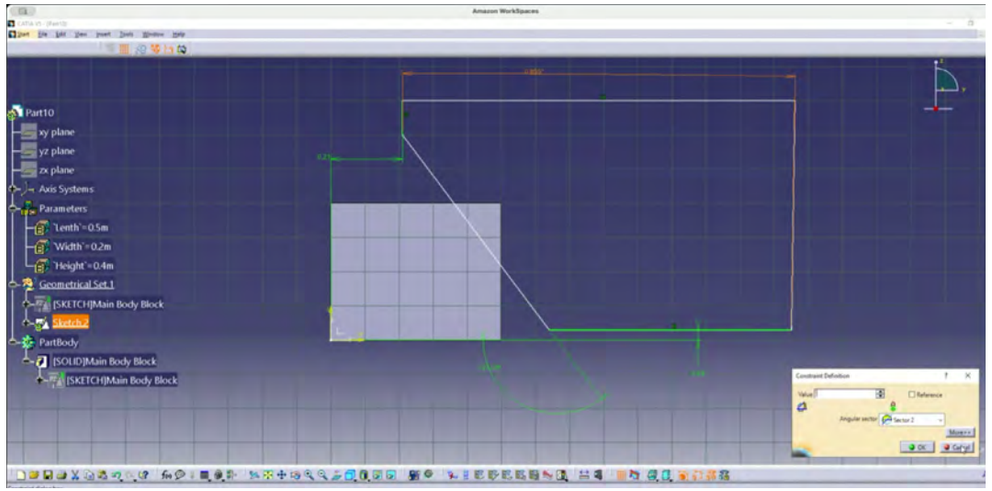
Example Level 2 - More organized CAD with clear labeling and thought towards collaboration with the design team, User defined parameters to drive model changes



LEVEL 3 - FOR HUMANS AND MACHINES (AUTOMATION PREP)

- Well organized CAD model
- Follows all best practices of the organization and DE Tool and data reuse techniques
- Organized Tree structure with clear labeling
- Many user defined parameters matching Requirements from other sources (like SysML or Requirements Tools)
- Utilizing formulas to correctly scale as needed
- Linking from other dependant models - Interface control geometry defined and references for single coherent Large Assemblies

Example Level 3 - More mature model utilizing company best practices, clear structure and labeling with global parameters and formula driven constraints, links to other models and design constraints for better situational awareness



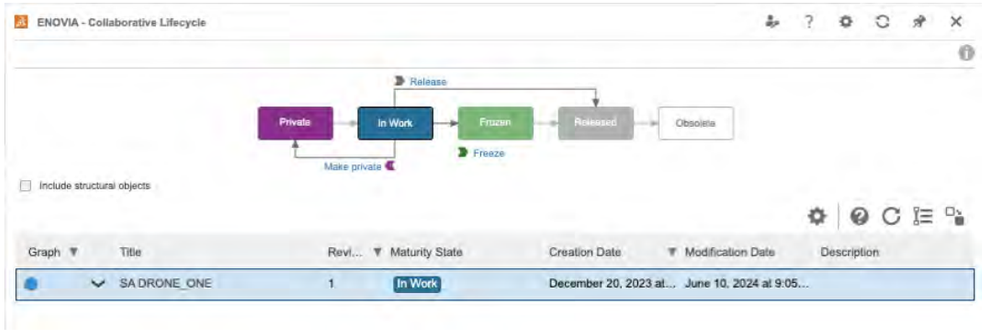
PLM - 3DEXPERIENCE EXAMPLES

LEVEL 1 - FOR ME

PLMs are for large teams to collaborate together on CAD and other documents, but if an inexperienced individual does not utilize the PLM properly, it can be detrimental.

- Builds CAD models (parts and assemblies) and saves on their local machine for too long which does not utilize the PLM features
- Once finally added to PLM, Checks in the model occasional and long intervals in between which inhibits others from collaborating on the project
- Design Reviews are done by screen shots and verbal explanations and some text inputs
- Bill of Materials (BOMs) have mistakes and inaccuracies to the needed product, takes more than expected effort to get right
- Engineering Change Notice process is unfamiliar and hard to follow, frequent rework of Change notices and takes much longer than expected
- No utilization of extra PLM features to make the process faster and more efficient

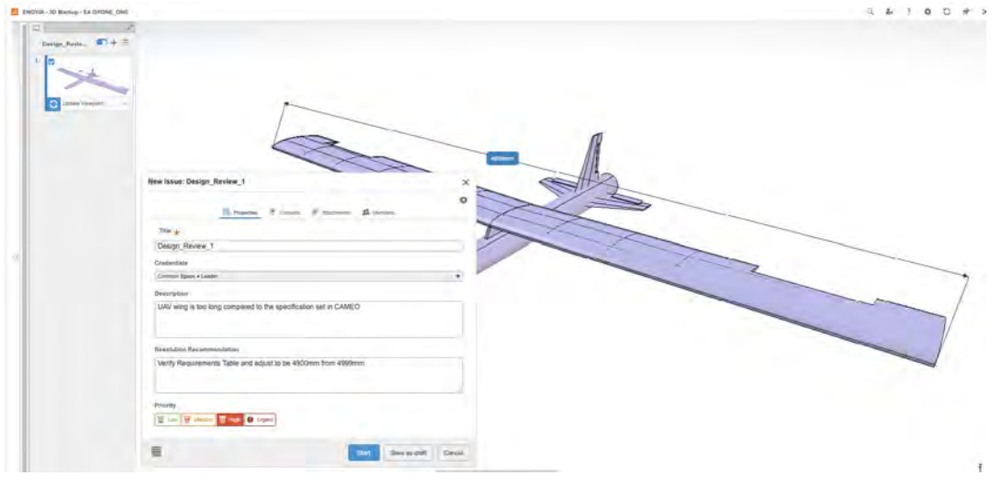
Example Level 1 - View of the change management process in 3DExperience Enovia



LEVEL 2 - FOR OTHER HUMANS

- As early as feasible, CAD models are saved into the PLM with full part numbers and applicable metadata loaded per part
- Check in CAD data regularly, at least daily to enhance team collaboration on the large Assemblies,
- Good version notes when checked in to give context to the updates
- Utilizes PLM Digital tools available to enhance design reviews to make up designs and share between users to collect feedback
- Robust Bill of Materials (BOMs), checked thoroughly and represents the needed product
- Engineering Change Notice process is familiar and Change Notices go through efficiently with minimal rework through the approval process
- Utilizes any features in the PLM available and trained on like Supply Chain or Quality Management features to attach quality plans or supplier information to products where applicable

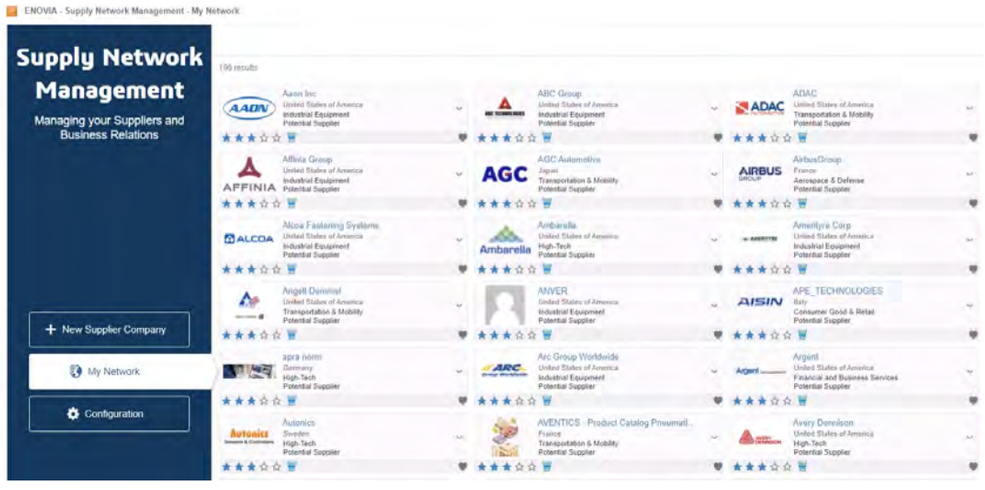
Example Level 2 - Utilizing the Design Review markup features in the PLM to enhance collaboration



LEVEL 3 - FOR HUMANS AND MACHINES (AUTOMATION PREP)

- CAD models are created from the PLM with full part numbers and applicable metadata loaded per part
- Check in CAD data regularly, possibly multiple times daily to maximize team collaboration on the large Assemblies sharing updates to the team often
- Thorough version notes when checked in to give context to the updates
- Utilizes PLM Digital tools available to enhance design reviews to make up designs and share between users to collect feedback
- Robust Bill of Materials (BOMs) auto populates from the Model Based Design (MBD) CAD which is a true representation of the product, checked thoroughly
- Expert on the Engineering Change Notice process and Change Notices go through efficiently with minimal rework through the approval process
- Utilizes any features in the PLM available and trained on like Supply Chain or Quality Management features to attach quality plans or supplier information to products where applicable
- Utilizes automation to streamline processes where possible

Example Level 3 - Utilizing features like Supply Chain Management tools in the PLM



HIGH FIDELITY FEM - MSC NASTRAN EXAMPLE

LEVEL 1 - FOR ME

- Creates a basic finite element model in MSC NASTRAN to analyze a single component or simple system.
- Uses default material properties without establishing a consistent material library.
- Mesh settings are generic, with minimal attention to mesh quality, refinement, or optimization.
- Boundary conditions, loads, and constraints are applied without thorough validation or systematic organization.
- Results are processed and reviewed directly in bulk output files, often lacking formatting for reuse or sharing.
- No systematic approach to model verification or validation, leading to potential inaccuracies in results.
- Focus is primarily on solving personal or isolated problems without preparing the model for broader collaboration or scalability.

Example Level 1 - Simple model, no modularity, minimal optimization, hard-coded values.

```
$ Simple Cantilever Beam Analysis - Single Component
SOL 101
CEND
BEGIN BULK

$ GRID Cards - Defining Nodes
GRID, 1, , 0.0, 0.0, 0.0
GRID, 2, , 1.0, 0.0, 0.0
GRID, 3, , 2.0, 0.0, 0.0

$ CBEAM Element - Single Beam
CBEAM, 1, 1, 1, 2, , 0.0, 1.0, 0.0

$ Material Definition
MAT1, 1, 210000.0, 0.3, 7.85E-09

$ Cross-Section Definition (Dummy Properties)
PBARL, 1, 1, RECT, 0.1, 0.1

$ Boundary Conditions - Clamped at Node 1
SPC1, 1, 123456, 1

$ Force at Free End
FORCE, 1, 3, , 0.0, -100.0, 0.0

ENDDATA
```

LEVEL 2 - FOR OTHER HUMANS

- Develops well-organized FEM models in MSC NASTRAN, adhering to organizational standards and best practices.
- Material properties are defined consistently and stored in a shared material library for use across the team.
- Mesh quality is optimized for accuracy and computational efficiency, with clear documentation of refinement strategies.
- Boundary conditions, loads, and constraints are carefully documented and consistently applied to ensure clarity and reproducibility.
- Results are shared in standardized formats, using visualization tools like Patran or HyperMesh, to support team decision-making.
- Verification and validation processes are followed to ensure the accuracy and reliability of results.
- Models are modular and structured, making it easier for other team members to understand, edit, or extend them.
- Utilizes NASTRAN's scripting and batch processing capabilities to automate repetitive tasks like load case variations or parameter sweeps.

Example Level 2 - Well-organized, modular structure, uses best practices, comments for clarity.

```
$ Cantilever Beam Analysis - Modular and Documented
SOL 101
CEND
BEGIN BULK

$ Grid Definition - Node Coordinates
$ Format: GRID, ID, CP, X, Y, Z
GRID, 1, , 0.0, 0.0, 0.0   $ Fixed end
GRID, 2, , 1.0, 0.0, 0.0

$ Element Definition
$ Format: CBEAM, EID, PID, GA, GB, SA, SB, W1A, W2A, W3A
CBEAM, 1, 1, 1, 2, , 0.0, 1.0, 0.0

$ Material Definition
MAT1, 1, 210000.0, 0.3, 7.85E-09   $ Steel: E=210 GPa, Nu=0.3, Density=7.85E-09

$ Beam Section Properties
$ Format: PBARL, PID, MID, SHAPE, DIM1, DIM2
PBARL, 1, 1, , RECT, 0.1, 0.1  $ Rectangular cross-section: 0.1m x 0.1m

$ Boundary Conditions
SPC1, 1, 123456, 1   $ Fixed end constraints

$ Load Application
FORCE, 1, 3, , 0.0, -100.0, 0.0   $ 100 N downward force at free end

$ Output Control
PARAM, POST, -1
PARAM, PRGPST, YES
PARAM, DISPLACEMENT, ALL
PARAM, STRESS, ALL

ENDDATA
```

LEVEL 3 - FOR HUMANS AND MACHINES (AUTOMATION PREP)

- Creates FEM models in MSC NASTRAN fully aligned with organizational best practices, designed for both human collaboration and machine automation.
- Material libraries are integrated with engineering tools like PLM systems to ensure seamless reuse and consistency across projects.
- Parameterized models are developed to enable rapid iteration and scalability for varying design scenarios.
- Automation scripts and batch jobs are used to perform parametric studies, sensitivity analyses, or optimization tasks efficiently.
- Fully documented workflows ensure seamless integration with the broader digital thread, linking NASTRAN models to CAD and MBSE tools like Cameo.
- Simulation data is structured for reuse in future analyses or downstream processes, such as digital twin creation or lifecycle management.
- Models are prepared for multiphysics coupling (e.g., thermal-structural or fluid-structural interactions) in a coordinated and automated way.
- Output data is formatted in machine-readable formats (e.g., CSV, XML) to facilitate integration with analytics platforms and automated reporting systems.
- Models support advanced capabilities like Monte Carlo simulations and probabilistic analyses to identify and mitigate worst-case scenarios, enhancing design robustness.
- The workflow is fully integrated into the organization's digital engineering ecosystem, ensuring traceability and scalability for future use.

Example Level 3 - Fully parameterized, reusable structure, automation-ready.

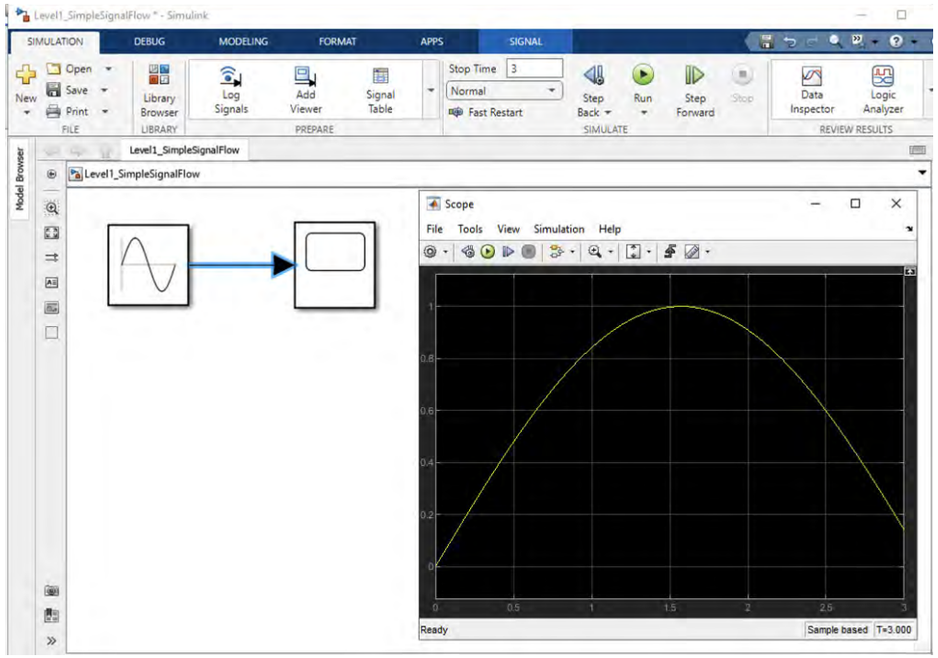
```
$ Parameterized Cantilever Beam Analysis - Automation Ready
SOL 101
CEND
BEGIN BULK

$ Parameters for Automation
PARAM, E_MODULUS, 210000.0   $ Young's modulus (MPa)
PARAM, POISSON, 0.3          $ Poisson's ratio
PARAM, DENSITY, 7.85E-09     $ Density (Mg/mm^3)
PARAM, BEAM_DIM1, 0.1        $ Beam width (m)
PARAM, BEAM_DIM2, 0.1        $ Beam height (m)
PARAM, FORCE_MAG, -100.0     $ Force magnitude (N)

$ Grid Definition
GRID, 1, , 0.0, 0.0, 0.0   $ Fixed end
GRID, 2, , 1.0, 0.0, 0.0

$ Element Definition
CBEAM, 1, 1, 1, 2, , 0.0, 1.0, 0.0

$ Material Definition - Parameterized
MAT1, 1, E_MODULUS, POISSON, DENSITY

$ Beam Section Properties - Parameterized

PBARL, 1, 1, , RECT, BEAM_DIM1, BEAM_DIM2

$ Boundary Conditions
SPC1, 1, 123456, 1    $ Fixed end constraints

$ Load Application - Parameterized
FORCE, 1, 3, , 0.0, FORCE_MAG, 0.0

$ Output Control
PARAM, POST, -1
PARAM, PRGPST, YES
PARAM, DISPLACEMENT, ALL
PARAM, STRESS, ALL

ENDDATA
```

LOW FIDELITY SIMULATION - MATLAB SIMULINK EXAMPLE

LEVEL 1 - FOR ME

- Develops a basic Simulink model for personal use to simulate a specific subsystem or problem.
- Models are not well-organized; block and subsystem names are generic or inconsistent.
- Parameters and variables are hard-coded within blocks, limiting reusability.
- Simulation focuses on answering a single question without considering future use or collaboration.
- No systematic testing or validation; results may be prone to errors or inconsistencies.
- Limited or no documentation, making it difficult for others to understand or use the model.
- Focus is solely on individual learning or solving a specific problem.

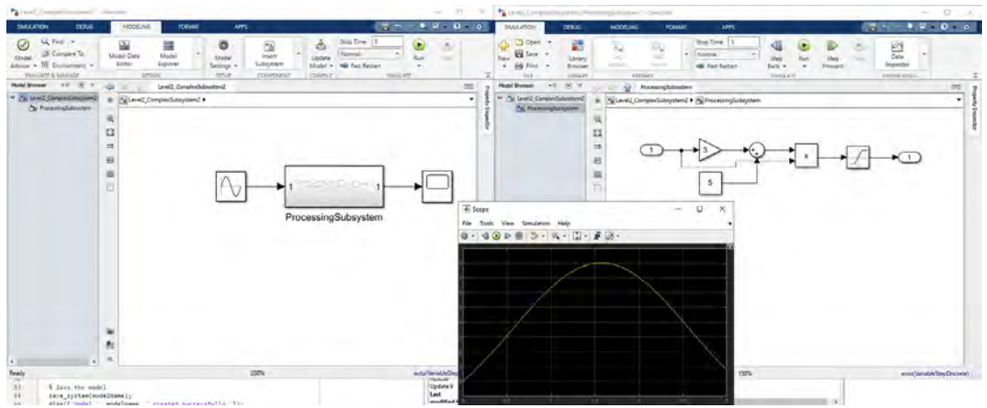
Example Level 1 - A basic simullink model to model a small specific subsystem



LEVEL 2 - FOR OTHER HUMANS

- Builds Simulink models that are modular, well-organized, and easy for others to understand.
- Clear naming conventions and hierarchical subsystems are used to improve model readability.
- Parameters and variables are stored in workspaces or configuration files, allowing for easier updates and reuse.
- Comprehensive testing and validation of the model ensure accurate and reliable results.
- Documentation and annotations are added to explain the purpose and functionality of each subsystem.
- Models are designed to handle multiple scenarios or use cases, enabling collaboration and iteration within a team.
- Utilizes Simulink tools like Stateflow, Simulink Test, or Simulink Coverage for enhanced analysis and robustness.
- Simulation results are shared in standardized formats with visualizations and reports for team-wide understanding.

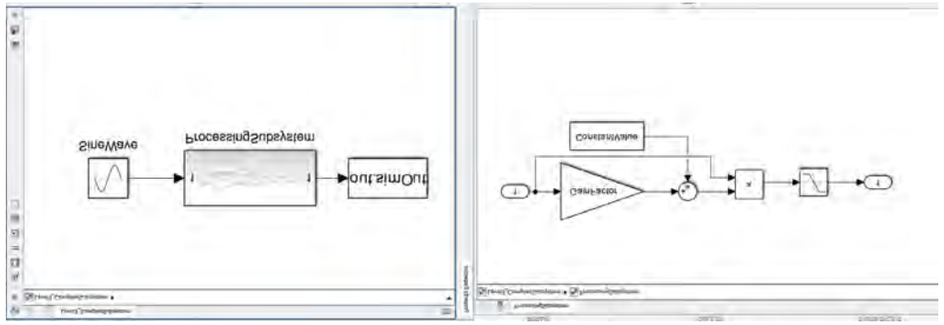
Example Level 2 - A larger organized model with subsystems, clear labels and scoped outputs



LEVEL 3 - FOR HUMANS AND MACHINES (AUTOMATION PREP)

- Creates Simulink models fully aligned with organizational best practices and prepared for automation and integration.
- Models are parameterized, scalable, and designed for flexibility across multiple projects or scenarios.
- Automated scripts or functions are used to run batch simulations, parameter sweeps, or optimization tasks.
- Links models with MATLAB scripts, data files, or external databases for seamless integration and real-time data updates.
- Incorporates modular design and reuse techniques, with shared libraries for subsystems and blocks.
- Simulation workflows are integrated with MBSE tools (e.g., Cameo or SysML) for traceability and alignment with system requirements.
- Supports co-simulation with other platforms (e.g., ANSYS, CAD tools) for complex multiphysics or system-level analyses.
- Simulation results are output in machine-readable formats (e.g., JSON, HDF5) for automated post-processing and reporting.
- Models are designed to support Monte Carlo simulations, probabilistic analyses, and real-time simulations for digital twin or hardware-in-the-loop (HIL) applications.
- Fully integrated into the organization's digital thread, enabling traceability from requirements to validation, with results feeding directly into downstream systems or tools.

Example Level 3 - Characteristics: This version incorporates advanced features like parameterization, automation for batch simulations, integration with external data, and outputs results in machine-readable formats. I've added comments and explanations to highlight the changes made to align with Level 3 standards.

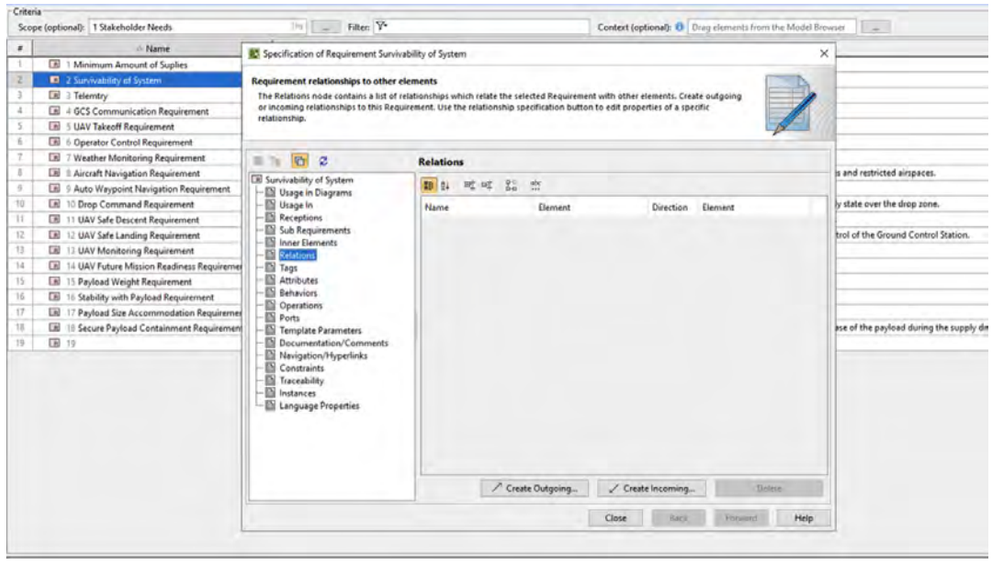


MBSE - CAMEO SYSTEMS MODELER EXAMPLE

LEVEL 1 - FOR ME

- Creates SysML diagrams in CAMEO to represent basic system requirements and structures.
- Diagrams are functional but lack organization or consistent notation.
- Minimal use of reusable components like blocks or libraries; no focus on shared data models.
- Limited Inconsistent traceability between requirements, logical models, and physical elements.
- Models serve only the individual's immediate needs with limited scalability for collaboration.

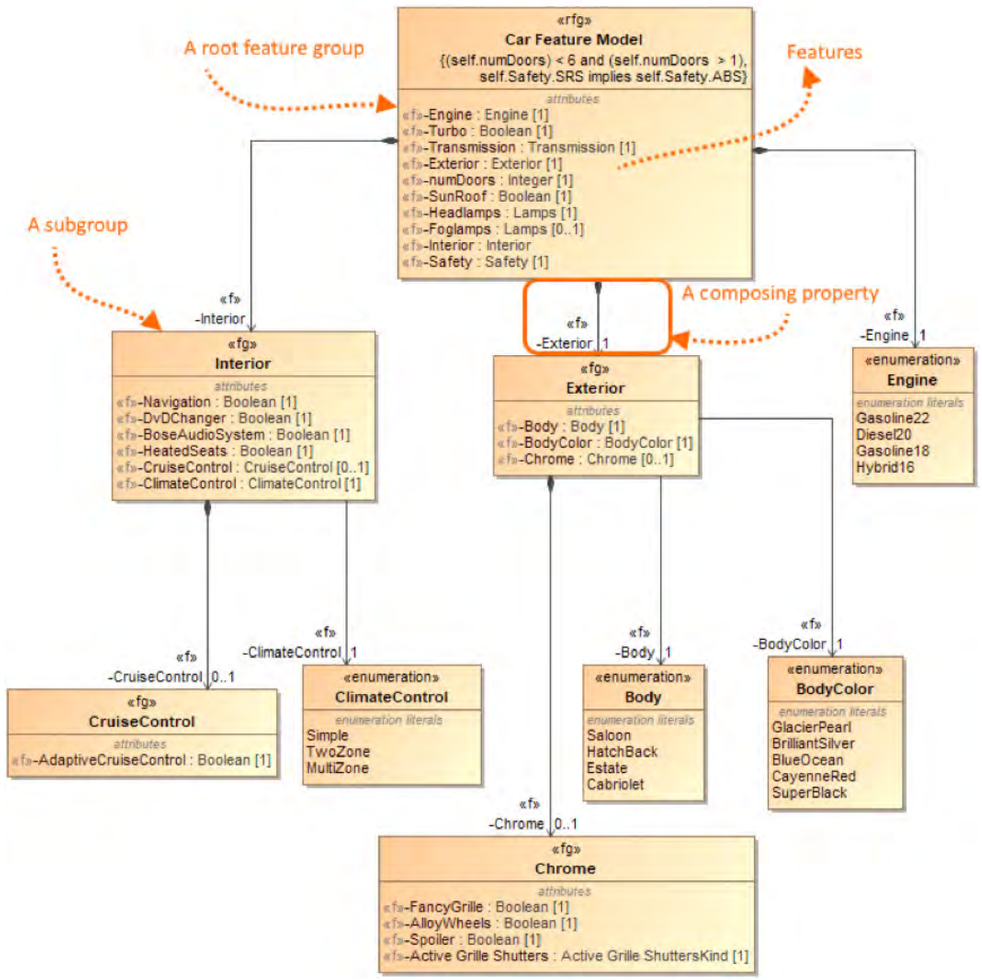
Example Level 1 - Requirements do not have relations to connect to dependents. Full depth is missing in this model for a complex system



LEVEL 2 - FOR OTHER HUMANS

- Models are structured and organized using best practices in SysML.
- Clear documentation and annotations allow others to understand and collaborate on the model.
- Elements such as blocks, activities, and states are linked to requirements for enhanced traceability.
- Shared libraries and reusable elements are used to ensure consistency across models.
- The model enables effective collaboration and review, including automated reporting of requirements coverage and traceability matrices.
- Validation rules are used to ensure model quality, and model reviews are conducted collaboratively.

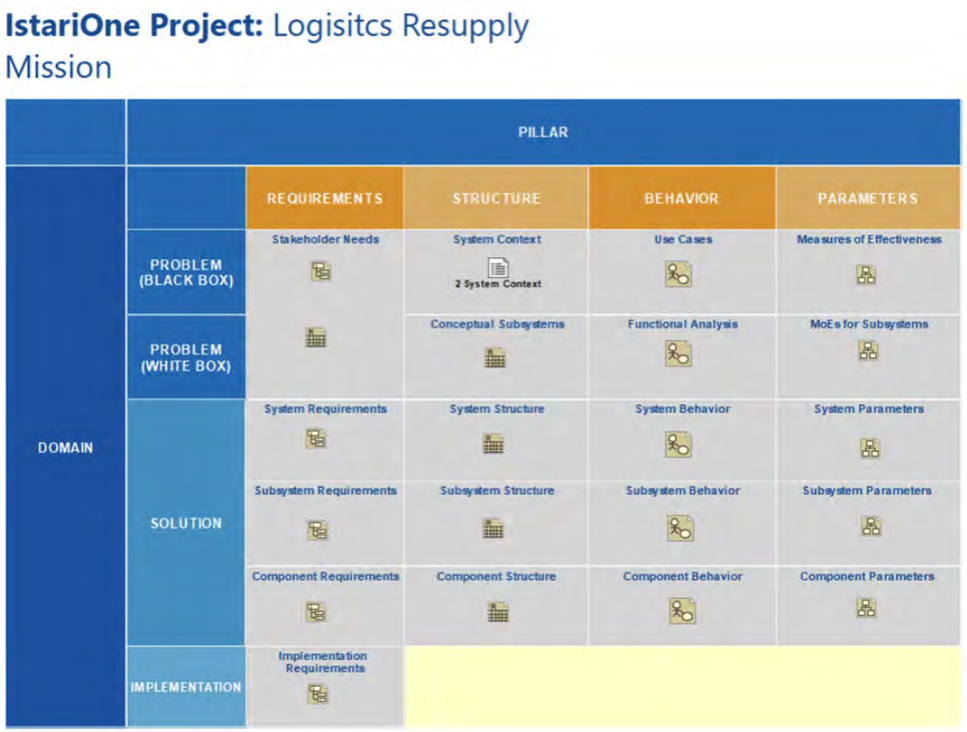
Example Level 2 - Detailed Feature model example from no magic documentation for well organized diagram and follows appropriate standards



LEVEL 3 - FOR HUMANS AND MACHINES (AUTOMATION PREP)

- Models adhere to all organizational and DE best practices, with a fully integrated digital thread.
- Requirements, functions, logical, and physical elements are fully traceable across the RFLP framework.
- Models are built with future automation in mind, using consistent naming conventions, reusable libraries, and parametric definitions.
- Logical models are prepared to inform downstream simulations, CAD, and other design tools through automated workflows.
- Custom scripts or plug-ins are used to automate routine modeling tasks and validations.
- Simulation-ready models are created to connect with tools such as MATLAB/Simulink, ensuring seamless integration for performance analysis.
- Data reuse is maximized by linking with external sources like requirements databases (e.g., DOORS), PLM systems, or CAD.
- The model is robust, scalable, and machine-readable, enabling efficient updates, scenario analysis, and integration into the organization’s broader digital engineering ecosystem.

Example Level 3 - Large Systems model will have organized Landing pages linking throughout the system and diagrams, even to sub system landing pages for large teams to collaborate. Linked to other models and simulations for a fully connected RFLP framework

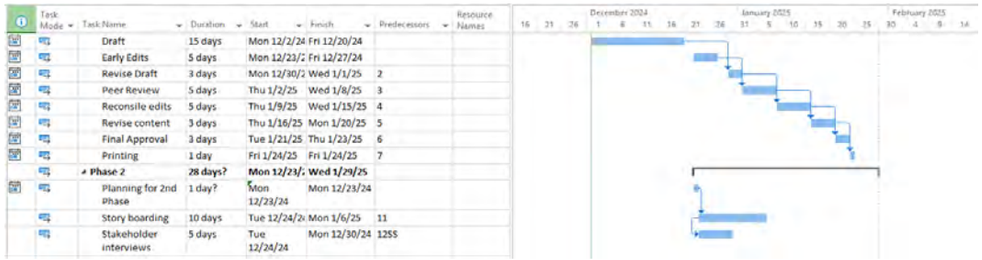


PRODUCTIVITY - MICROSOFT PROJECT EXAMPLE

LEVEL 1 - FOR ME

- Creates a simple project schedule in MS Project for personal tracking.
- Tasks, durations, and dependencies are entered manually without consistent formatting.
- Limited use of task hierarchies or milestones; focuses primarily on individual tasks.
- No use of resource allocation or tracking; assumes all work is self-contained.
- Updates are performed sporadically, making the schedule unreliable for broader use.
- No documentation or notes to explain task details, priorities, or rationale.
- Focus is solely on managing personal workload without considering team collaboration.

Example Level 1 - A simple project gantt chart for one person to track a project

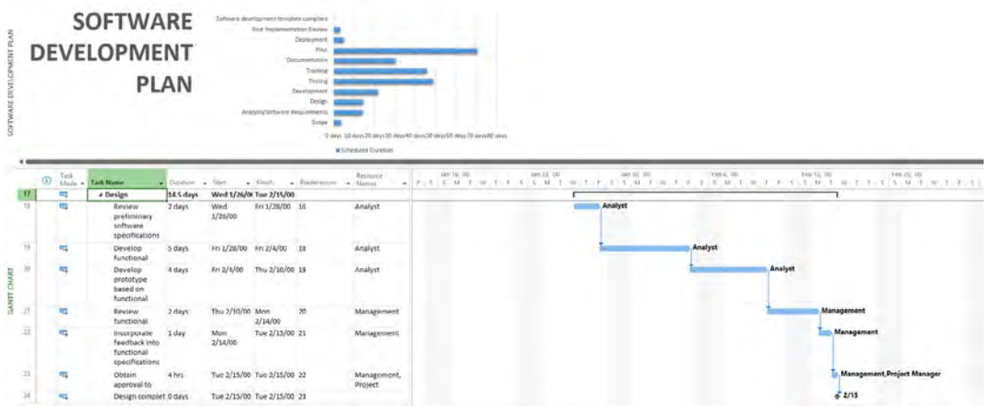


Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
	Draft	15 days	Mon 12/2/24	Fri 12/20/24		
	Early Edits	5 days	Mon 12/23/2	Fri 12/27/24		
	Revise Draft	3 days	Mon 12/30/2	Wed 1/1/25	2	
	Peer Review	5 days	Thu 1/2/25	Wed 1/8/25	3	
	Reconsile edits	5 days	Thu 1/9/25	Wed 1/15/25	4	
	Revise content	3 days	Thu 1/16/25	Mon 1/20/25	5	
	Final Approval	3 days	Tue 1/21/25	Thu 1/23/25	6	
	Printing	1 day	Fri 1/24/25	Fri 1/24/25	7	
	Phase 2	**28 days?**	**Mon 12/23/**	**Wed 1/29/25**		
	Planning for 2nd Phase	1 day?	Mon 12/23/24	Mon 12/23/24		
	Story boarding	10 days	Tue 12/24/2	Mon 1/6/25	11	
	Stakeholder interviews	5 days	Tue 12/24/24	Mon 12/30/24	12SS	

LEVEL 2 - FOR OTHER HUMANS

- Develops well-structured project plans with clearly defined task hierarchies, milestones, and dependencies.
- Uses consistent formatting and naming conventions to improve readability and understanding.
- Resources are allocated and tracked, including effort, costs, and availability for team members.
- Task progress is updated regularly, ensuring the schedule remains current and accurate.
- Shared project files allow team members to view progress and understand dependencies.
- Incorporates annotations and documentation for key tasks and milestones to provide context for stakeholders.
- Leverages built-in MS Project features like critical path analysis and baseline comparisons for decision-making.
- Utilizes reporting features to share project status, risks, and upcoming deliverables with stakeholders.

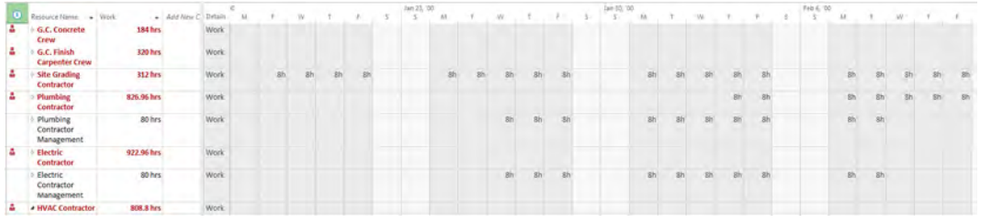
Example Level 2 - Large well structured project file with resourcing and costs as well as reports to align with other stakeholders



LEVEL 3 - FOR HUMANS AND MACHINES (AUTOMATION PREP)

- Creates a fully integrated project management plan that aligns with organizational best practices and supports automation.
- Projects are linked to shared resource pools, enabling organization-wide resource optimization.
- Task dependencies are dynamically updated based on external changes, ensuring real-time accuracy.
- Automates repetitive tasks like recurring updates, status reporting, and notifications using MS Project macros or Power Automate.
- Integrates MS Project with other tools such as PLM systems, ERP software, or requirements management tools to maintain a single source of truth.
- Data-driven updates ensure that project plans remain consistent with live input from other systems or teams.
- Custom dashboards and reports provide real-time insights for stakeholders, including predictive analytics for potential delays or risks.
- Models are designed to support Monte Carlo simulations or risk analyses for schedule optimization and contingency planning.
- Fully integrated into the organization's digital thread, with project data contributing to overall portfolio management and strategic decision-making.

Example Level 3 - A full in depth project model that fully utilizes the features of the tool to not only collaborate with the larger team but also ready to integrate with other tools



The Culture of The Digital Engineering Organization

Building a successful digital engineering organization requires more than just tools, processes, and technologies—it requires a culture that fosters collaboration, innovation, and continuous improvement. The culture of a digital engineering organization is the foundation upon which its people, tools, and workflows thrive, enabling the creation of complex systems and transformative solutions.

COLLABORATION ACROSS DISCIPLINES

At the heart of this culture is collaboration. Digital engineering thrives when individuals and teams from diverse disciplines—mechanical, electrical, software, systems engineering, and beyond—work seamlessly together. The digital-engineering culture must encourage open communication, and break down silos.

DATA AS A SHARED ASSET

A defining aspect of this culture is treating data as a shared asset rather than a siloed resource. Engineers in a digital organization recognize the importance of producing data that is reusable, interpretable, and valuable to others. This requires a mindset shift where individuals see their work not as isolated outputs but as critical inputs into a larger ecosystem. Where engineers do the extra work to label specific data with the right level of classification which will enable other people to use parts of the data, rather than simply labeling the whole document as a single level of classification which will eliminate vast groups from accessing information that they would have otherwise been able to access.

EMBRACING AGILITY AND CONTINUOUS LEARNING

The culture of a digital engineering organization must embrace agility, encouraging teams to iterate quickly and adapt to changes in technology, requirements, and business needs. A software development skillset will be required for the digital-engineer and they will need to grow their abilities while also growing their core competencies. Continuous learning is a cornerstone of this culture, with engineers regularly upskilling to stay ahead of industry advancements. Leaders must champion curiosity, experimentation, and learning from both successes and failures.

A SYSTEMS THINKING MINDSET

In a digital engineering culture, systems thinking is a core value. Engineers understand the interconnected nature of their work and strive to see the bigger picture. They design solutions that are not just optimal for their discipline but integrate seamlessly into the broader system, balancing technical performance with business goals.

LEADERSHIP THROUGH VISION AND EMPOWERMENT

Leadership in the digital-engineering culture extends beyond managing projects—it's about inspiring teams with a clear vision, providing the resources and support needed to innovate, and empowering individuals to take ownership of their contributions. Leaders prioritize creating an environment where engineers feel confident to take risks and make impactful decisions.

GOVERNANCE AND STANDARDIZATION

While innovation thrives on creativity, a strong digital engineering culture balances this with governance and standardization. Consistent processes, shared best practices, and clear guidelines ensure that engineers can build on each other's work effectively. This structured approach eliminates unnecessary friction and amplifies productivity across teams.

CELEBRATING SUCCESSES AND DRIVING PURPOSE

Finally, a thriving digital engineering organization celebrates its successes and connects its engineers to a larger purpose. Whether it's solving a global challenge or delivering a groundbreaking product, engineers are motivated when they see the tangible impact of their work. Leaders cultivate a sense of pride and purpose, reminding teams of the value their innovations bring to the world.

5

Looking Ahead: From Fantasy To Reality

As we look to the horizon of digital engineering, what once seemed like fantasy becomes an ever-closer reality. The transformation outlined in these pages—akin to the shift from lone virtuoso performers to a well-coordinated orchestra—is already underway. Organizations that embrace this transition will find themselves in a new era of innovation, one where the barriers of time, cost, complexity, and organizational silos dissolve into a continuous flow of creativity and execution.

THE ENGINEER'S CHANGING WORLD:

Tomorrow's engineering ecosystem will be defined by immersive virtual environments that allow designers, analysts, and integrators to collaborate in real time, no matter where they are in the world. Models will not just represent systems; they will become systems—living, dynamic entities updated continually with operational and test data. Engineers will tweak a parameter on a digital twin and watch changes ripple through simulations, procurement schedules, supply chain forecasts, and certification documents in seconds. Where today we rely heavily on physical prototypes and lengthy design cycles, tomorrow's engineers will refine designs virtually and, by the time a physical prototype is built, it will be almost production-ready.

FROM FRAGMENTED TOOLSETS TO UNIFIED PLATFORMS:

The engineering toolchains of the future will integrate seamlessly. Rather than juggling proprietary formats, today's complicated stacks of CAD files, simulation results, and spreadsheets will give way to an interconnected, AI-assisted ecosystem. Tools will become modules—interoperable "instruments" that can be dynamically connected. An MBSE environment might trigger simulations based on requirements changes, automatically calling upon CAD to adjust geometry, PLM to update part lifecycles, or mission modeling tools to re-evaluate system performance at scale. The engineer will orchestrate these connections—like a maestro—guiding the digital thread rather than wrestling with it.

AI, AUTOMATION, AND THE HUMAN FACTOR:

Artificial intelligence will increasingly act as a quiet collaborator, augmenting human abilities at all levels. By identifying patterns across historical projects, AI agents will propose solutions, fill in gaps, and highlight risks before human experts even realize they exist. Automated workflows will run day and night, exploring vast parametric design spaces and converging on optimal solutions. Yet, the human engineer will remain at the center—providing creativity, intuition, leadership, and judgment. Engineers will set goals, interpret subtle requirements, and ensure that the organization's human values—safety, ethics, sustainability—are woven into every decision. Humans and machines will form a complementary partnership, enabling levels of innovation previously unthinkable.

CROSS-DISCIPLINARY FLUENCY AND NEW ROLES:

Just as the boundaries between mechanical, electrical, and software engineering have blurred, so too will the boundaries between engineering and fields like computer science, data analytics, and user experience design. The "engineer of the future" will need a broader perspective, understanding how choices in one domain echo through the entire system. New roles will emerge: data curators who ensure information flows smoothly, digital-thread integrators who connect tools and teams, and AI ethicists who ensure that automated decision-making aligns with organizational principles. Training and education will adapt, emphasizing systems thinking, model-based approaches, and coding literacy right alongside traditional engineering fundamentals.

BUILDING TRUST AND CONFIDENCE AT SCALE:

Perhaps most crucially, this new reality will enable organizations to build greater confidence in their solutions. By simulating scenarios thousands of times before any metal is cut, by verifying requirements continuously, and by linking test data back into models, organizations will arrive at fully vetted, production-ready designs far faster than today's methods allow. This will not only shrink development cycles and reduce costs but also raise the bar on quality, safety,

and reliability. It will allow companies to push the envelope, trying bold new concepts with less risk and greater agility, advancing technology at a speed once considered unattainable.

THE PATH FORWARD:

None of this will happen overnight. Cultural changes are needed to support data sharing, transparency, and cross-disciplinary teamwork. Infrastructure investments must be made to ensure the proper IT environment, cybersecurity measures, and computational resources. Training, professional development, and mentorship will be required to turn today's specialists into tomorrow's maestros. But the prize is clear: organizations that can combine human ingenuity with orchestrated toolsets and intelligent automation will lead in their markets, industries, and domains.

A NEW GOLDEN AGE OF ENGINEERING:

We stand at the cusp of a new golden age of engineering, where the impossible slowly turns into the achievable. The transition from soloists to orchestras, from fragments to threads, from physical-first to digital-first is a profound shift—one that will redefine the very nature of engineering. As you embark on this journey, remember the analogy of the maestro and the orchestra. The conductor does not play every instrument, yet they bring together specialists who collectively produce something far greater than any could alone. In the same way, your organization can learn to harmonize diverse disciplines, tools, and methodologies to create bold new innovations—turning the visions of the future into today's engineered reality.

32.5
30
27.5
25
22.5
20
17.5
15
12.5
10
7.5
5
2.5
Vorticity Magnitude (Hz)

0.3
0.275
0.25
0.225
0.2
0.175
0.15
0.125
0.1
0.075
0.05
0.025
0
Cf