From Art to Engineering: The Birth of Software Engineering during the Development of the Apollo Guidance Computer (1961-1971)

https://tinyurl.com/birth-of-softwareengineering

Rory Ward, B.Sc.

roryward@gmail.com

University College Dublin

College of Arts and Humanities

This dissertation is submitted in part fulfilment of the Master of Arts in Global History

Submitted: July 2025

Updated: Oct 2025

Head of School: Professor Catherine Cox

Supervisor: Professor Alexander Wilkinson

Abstract

Software Engineering is used across the world to build software that we interact with every day. The origins of this discipline go back to the 1960s when the Massachusetts Institute of Technology (MIT) developed the Apollo Guidance Computer to meet President John F. Kennedy's commitment to "achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to the earth".[1] During the period at the beginning of the digital computer age, when computers were the size of rooms and programming consisted of punching holes in cards, MIT built the first small, real time, interactive computer that controlled the spacecraft that landed men on the Moon. This technical marvel was a key component in the Apollo program with responsibility for the navigation and guidance functions of the Moon missions. The teams at MIT, which included Margaret Hamilton, developed the software for this computer and in the process invented the term Software Engineering and the processes that make up the discipline. Using extensive primary sources from MIT and NASA, this dissertation charts the development of the software for the Apollo Guidance Computer and how it gave rise to the birth of Software Engineering as an engineering discipline.

[1] Yanek Mieczkowski, *Eisenhower's Sputnik Moment: The Race for Space and World Prestige*. 1st ed. Ithaca: Cornell University Press, 2013, 265

Table of Contents

Introduction *1*

1 Genesis and Early Software Development Stages (1961–1964) *8*

3 Initial Inroads into Software Engineering (1964–1966) *21*

4 Crisis and the Emergence of Software Engineering (1966–1968) *26*

5 Software Engineering Matures (1968–1971) *39*

Conclusion *43*

Appendix: Apollo Guidance Computer Software Releases *47*

Bibliography *49*

Primary Sources **49**

Secondary Sources **58**

Introduction

Computers are ubiquitous in our modern world, but it is the software that runs on them that dictates how these computers behave.[2] Everything from financial systems to manufacturing processes to household appliances and everything in between uses computer software that is designed, written, tested, released, maintained and eventually retired. Modern life would be very different if we did not have the software that we interact with numerous times daily.

Software has two main branches: Computer Science and Software Engineering. Computer Science is the study of algorithms, data structures, computer architecture, network design and artificial intelligence. Software Engineering is defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software, and the study of such approaches".[3] In simpler terms, it is bringing visibility and predictability into software development by providing clarity on what features are required, how they will be implemented, when they will be delivered, whether they reliably work as expected and how much they will cost. This dissertation focuses on the historical foundations of Software Engineering during the Apollo Moon landing program.

The historiography of the Apollo missions is broad. A review of book length studies by Launius from 2006 divided the work into five areas of research: precursor foreign and public policy, technological innovation, the missions and cult of the astronaut, lunar science and the cultural impact of Apollo.[4] The first two of these areas are relevant to this dissertation. As of 2006, according to Launius, the most well-developed area was on policy analysis centring on the reasons behind the Space Race and the race to the Moon. Logsdon's *The Decision to Go to the Moon* from 1970 details the process that led Kennedy to commit the USA to landing a man on the moon.[5] This was in large part based on the seemingly unending list of humiliations imposed on the USA by USSR firsts in the space frontier, but also as a distraction to the recently failed Bay of Pigs

---

[2] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society* s2-42, no. 1 (1937): 230-265; Computers are Universal Turing Machines and software is the set of instructions to a Universal Turing Machine.
[3] *610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology*. S.l.: IEEE, 1990, 67
[4] Roger D. Launius, "Interpreting the Moon Landings: Project Apollo and the Historians." *History and Technology* 22, no. 3 (2006): 225-255.
[5] John M. Logsdon. *The Decision to Go to the Moon: Project Apollo and the National Interest.* Cambridge, MA: The MIT Press, 1970.

invasion of Cuba. Kennedy needed the USA to compete with the USSR in a way that allowed the USA to catch up and surpass their rivals and the audacious plan of landing on the moon was a clearly focused goal that could achieve that. McDougall's ... *The Heavens and the Earth: A Political History of the Space Race* from 1985 focuses on the rivalry between the USA and the USSR between the mid 1950s and the mid 1960s that led to the Space Race.[6] He argued that the Space Race institutionalized a 'perpetual technological revolution' where states were the instigator of technological innovation, with Apollo being the main example. Since Launius' review, Mieczkowski's *Eisenhower's Sputnik Moment: The Race for Space and World Prestige* from 2013 focuses on President Eisenhower, who was in office when Sputnik was launched in 1957, rather than on President Kennedy.[7] While detailing Eisenhower's positive contributions, such as the strengthening of the USA's defensive missile capabilities and the creation of a civilian NASA organisation, Mieczkowski also points to Eisenhower's underestimation of the damage to the USA's domestic and international prestige due to the USSR's successes in the early days of the Space Race.

Launius' second area of historiography is on technology innovation. The paper credits the NASA History Series for detailing a significant part of this technology and gives Bilstein's *Stages to Saturn* from 1980 as a notable example from this series that details the creation of the Saturn V rocket.[8] Also mentioned is *Apollo: Race to the Moon* from 1989 that focuses on the managers, engineers and processes that ran the whole program.[9] For the purpose of this dissertation, also referenced is Hall's *Journey to the moon: the history of the Apollo guidance computer* from 1996, which details the creation of the Apollo Guidance Computer.[10] The author was one of the principal designers of the computer and this monograph mainly focuses on the hardware aspects and does not go into significant detail on the software side. Missing from Launius's review is Tomayko's *Computers in spaceflight: the NASA experience* from 1988 which details how computers were used across many NASA missions, including Apollo and includes a good section on how NASA and MIT attempted to manage the process of creating the Apollo Guidance Computer hardware and

---

[6] Walter A. McDougall ... *The Heavens and the Earth: A Political History of the Space Age*. New York: Basic Books, 1985.
[7] Mieczkowski, *Eisenhower's Sputnik Moment*
[8] Roger E. Bilstein. *Stages to Saturn: A Technological History of the Apollo/Saturn Launch Vehicles.* Washington, DC: NASA SP-4206, 1980.
[9] Charles Murray and Catherine Bly Cox. *Apollo: The Race to the Moon*. New York: Simon and Schuster, 1989.
[10] Eldon C. Hall, *Journey to the moon: the history of the Apollo guidance computer*. Aiaa, 1996.

software.[11] Mindel's 2008 work *Digital Apollo: Human and Machine in Spaceflight* is an important work that details how the astronauts worked with the Apollo Guidance Computer and how the combination of human and computer provided the flexibility to ensure success in the moon landings.[12] It provides good detail on how the computer was used, with less detail on how the computer software was implemented. O'Brien's *The Apollo guidance computer: Architecture and operation* from 2010 builds upon Hall's and Mindel's work but focuses on the design and use of the computer, and less on the processes used to build the software.[13] None of the existing historiography focuses on the processes used to create the software that ran on the Apollo Guidance Computer and that guided man to the Moon.

A newer area of research has been on gender studies during the Apollo program. There were relatively few female leaders or engineers working in NASA or MIT during Apollo and more research is needed here. Shetterly's *Hidden Figures: The American Dream and the Untold Story of the Black Women Mathematicians Who Helped Win the Space Race* from 2016 tells the story of black women in the early days of NASA, focusing on Dorothy Vaughan who was the first black manager in the precursor to NASA, the National Advisory Committee for Aeronautics, and on the engineers Katherine Johnson and Mary Jackson.[14] Margaret Hamilton was an important figure in the creation of the Apollo Guidance Computer software, but was a relatively unknown figure until the 1990s. Maurer's *The Woman in the Moon: How Margaret Hamilton Helped Fly the First Astronauts to the Moon* from 2023 is a welcome addition to this space.[15] Another important contribution that attempts to broaden the research to the pool of skilled women who performed one of the key hidden production roles in the Apollo Guidance Computer is Shorey and Rosner's *A voice of process: re-presencing the gendered labor of Apollo innovation* from 2019. This research tells the forgotten story of women weavers who worked for Raytheon and who weaved the core rope memory that encoded all the software for the computer.[16]

---

[11] James E. Tomayko, "Computers in spaceflight: the NASA experience." *Kent, Allen; Williams, James G., eds. Encyclopedia of Computer Science and Technology* 18, no. NAS 1.26: 182505 (1988)
[12] David A. Mindell. *Digital Apollo: Human and Machine in Spaceflight*. 1st ed. Cambridge, MA: MIT Press, 2008;2011
[13] Frank O'Brien, ed. *The Apollo guidance computer: Architecture and operation*. New York, NY: Praxis, 2010.
[14] Margot Lee Shetterly. *Hidden Figures: The American Dream and the Untold Story of the Black Women Mathematicians Who Helped Win the Space Race.* New York: HarperCollins, 2016.
[15] Richard Maurer, *The Woman in the Moon: How Margaret Hamilton Helped Fly the First Astronauts to the Moon*. Roaring Brook Press, 2023
[16] Samantha Shorey and Daniela K. Rosner, "A voice of process: re-presencing the gendered labor of Apollo innovation." *communication+ 1* 7, no. 2 (2019).

Software Engineering as a standalone discipline has a history that goes back to the 1960s. However, the historiographical record is slim. Campbell-Kelly's *Computer: A History of the Information Machine* from 1996 credits NATO conferences in 1968 and 1969, where a set of contemporary experts discussed problems in software development, as an important milestone in the creation of Software Engineering.[17] O'Regan's *Brief History of Computing* from 2018 credits the same NATO conferences.[18] Also from 2018, Booch's *The History of Software Engineering* credits Margaret Hamilton, who worked on the Apollo Guidance Computer at the Massachusetts Institute of Technology (MIT), with inventing the term Software Engineering. None of the existing historiography focuses on the creation of the Software Engineering discipline during the Apollo program.

As defined today, Software Engineering provides an engineering focused approach to the development of software, which is applied throughout the Software Development Life Cycle of a software project. Although it is a broad discipline and under active research and ongoing development, it generally includes several common functions.

Requirements Analysis and Definition involves documenting in detail what the software is expected to accomplish. This usually involves a prioritization or phasing of different features. The purpose of this effort is to get agreement, and sign-off, on what is needed from the software, from all stakeholders.

Design and Architecture involves documenting how the software will be implemented, what technology will be used, what interfaces will be provided by the software, what other systems does the software need to interact with and how storage and/or networking will be implemented.

Implementation is the process of building the software and includes language choice and tooling choices. It can also include standards for version control and coding style. This is to provide a common basis for how the software is to be coded and to improve the future maintainability of the software.

---

[17] Martin Campbell-Kelly, *Computer: A History of the Information Machine*. 3rd ed. Boulder: Westview Press, 2014
[18] Gerard O'Regan, *Brief History of Computing*. 3;3rd 2021;Third; ed. Cham: Springer, 2021., 202

Testing Strategy defines the approach to testing the software to ensure that it meets the functional requirements. This can include testing tooling, test plans and requirements for different types of testing, such as unit testing, integration testing, performance testing and reliability testing.

Release Management defines how the software is to be released to customers.

User Acceptance defines how the requester of the software approves and agrees that the software meets their needs.

Software Maintenance involves the strategy of how to update software once it has been released and usually includes a tightly managed change control process. This process defines and approves what changes can be made. The final part of *Software Maintenance* is obsoleting the software, which can include migration to a new system.

Overseeing all these functions is a project schedule that tracks timelines and costs.

The gap in the historiography is the connection between the history of Software Engineering and the development of the software for the Apollo Guidance Computer. The focus of this original research is to investigate how the teams at MIT, including the team led by Margaret Hamilton, created the discipline of Software Engineering before the NATO conferences of 1968 and 1969 or whether they were just the originators of the term. We will be looking for documented evidence of Software Engineering practices, such as: requirements gathering, design, implementation, testing, release management and maintenance.

We will show, based on the available primary sources, that Software Engineering principles did not exist as the Apollo Guidance Computer was in its early days in 1961-1964, but that pressure to deliver high quality, reliable, “man-rated” software to enable the successful manned landings on the Moon before 1970 effectively forced the development of what we would recognize as Software Engineering principles by 1968.[19]

The main repository for primary sources is the Apollo and Gemini Document Library on The Virtual AGC Project website.[20] Generally, they are contemporary documents with small

[19] “man-rated” means that astronauts’ lives were at stake.
[20] “Apollo and Gemini Document Library”, The Virtual AGC Project, accessed 23 July 2023, https://www.ibiblio.org/apollo/links2.html#gsc.tab=0

distributions. The outlier to this is the five-volume *R-700, MIT's Role in Project Apollo*, which was written by MIT between 1971 and 1972 and can be assumed to have a MIT bias.[21] There may be further sources, such as early design documents, available at MIT, but these were unavailable.

Chapter 1, *Genesis and Early Software Development Stages (1961–1964)*, provides the political and technological landscape at the beginning of the Apollo program that led to the Apollo Guidance Computer project. It also introduces Margaret Hamilton up until she joined the MIT Instrumentation Laboratory as a computer programmer. Lastly, it provides a summary of the development of the computer hardware and describes the first few years of software development, when there was no Software Engineering in place.

The *Initial Inroads into Software Engineering (1964–1966)* chapter outlines the middle years of development when the first aspects of Software Engineering started to develop. NASA made the first attempt at defining a process for managing large software projects. MIT made improvements in documentation and testing and the Apollo Guidance Computer had its first unmanned launch. However, this wasn't enough to avert a crisis around delivery.

Crisis and the Emergence of Software Engineering (1966–1968) details the crisis years of 1966 and 1967 related to the lack of process and delays in the program. By mid to late 1967, this resulted in the eventual emergence of Software Engineering practices that helped resolve the crisis. A new organizational structure and process was put in place in MIT. NASA and MIT agreed on a process for defining, testing, accepting and changing software releases. Also, during this period, Margaret Hamilton coined the term "Software Engineering".

Finally, *Software Engineering Matures (1968–1971)* describes the final years of the program when MIT successfully delivered many versions of the software and the Apollo Guidance Computer successfully guided the manned missions to the Moon. It also reviews the motivations and outcomes of the NATO Software Engineering conferences in 1968 and 1969 and argues that MIT

---

[21] James A. Hand, "R-700, MIT's Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume I, Project Management, Systems Development, Abstracts and Bibliography", MIT Charles Stark Draper Laboratory, October 1971; Anonymous, "R-700, MIT's Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume II, Optical, Radar and Candidate Subsystems", MIT Charles Stark Draper Laboratory, March 1972; Eldon C. Hall, "R-700, MIT's Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume III, Computer Subsystem", MIT Charles Stark Draper Laboratory, August 1972; R-700, MIT's Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume V, The Software Effort", MIT Charles Stark Draper Laboratory, March 1971

and NASA had already experienced the issues raised at these conferences and used Software Engineering to resolve them. However, the learnings from the development of the Apollo Guidance Computer were not widely disseminated.

1 Genesis and Early Software Development Stages (1961–1964)

In this chapter, we will provide the political and technology context of the early 1960s and cover the creation of the Apollo Guidance Computer project at MIT. We will show that Software Engineering, as described in the Introduction, did not exist during this period.

The main driver of growth in computing after World War II was the Cold War between the USA and the USSR. Specifically, computers were required for the detailed computations needed for the development of nuclear weapons and the offensive and defensive capabilities of both superpowers. One of the first recognised computers was ENIAC which was built by the University of Pennsylvania between 1943 and 1946 for nuclear weapons research and for calculating shell firing tables. It replaced skilled human computers, usually women with mathematical backgrounds, who calculated trajectories by hand.[22] It was a huge machine, occupying a large room. It was programmed by physically setting hundreds of switches and connecting cables to different parts of the machine. It had 18,000 vacuum tubes, consumed 120,000 watts of power and could do 5,000 additions per second.[23]

These computers helped design Intermediate Range Ballistic Missiles (IRBMs) and Inter-Continental Ballistic Missiles (ICBMs), along with the nuclear warheads that these missiles carried. These missiles gave both superpowers the ability to launch nuclear weapons without the need for bombers. A new front of the Cold War opened on 4 October 1957, when the USSR launched Sputnik I, the world's first artificial satellite. This was the opening salvo of the Space Race, which was a race of propaganda and technological development. It was a Prestige Race between the two superpowers, which was used to showcase the technical prowess, and hence strength of the political philosophy, of each nation. There were many Soviet firsts during the early part of the Space Race: first artificial satellite, first living animal in space, first spacecraft to hit the Moon, first photographs of the back side of the Moon, first man in space, first woman in space and the first spacewalk. President Eisenhower viewed the Soviet achievements as purely a propaganda exercise, with limited value when it came to the defence strategy of the USA. He underestimated

[22] Jennifer S. Light, "When Computers were Women." *Technology and Culture* 40, no. 3 (1999): 455-483
[23] Hall, *Journey to the moon*, 31

the prestige aspect of these achievements, and he did not understand the reputational damage to the USA because of these USSR firsts, both at home and abroad. This changed when Kennedy came to power in 1961. At his State of the Union speech on 25 May 1961, six weeks after the USSR's Yuri Gagarin became the first man in space and a mere three weeks after the USA's first manned suborbital flight, flown by Alan Shepard, he committed the USA to "achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to the earth".[24] This was an incredibly audacious goal, especially since man had just barely gone into earth orbit for the first time and a lot of the required technology to accomplish Kennedy's goal did not yet exist. However, it did galvanise the public's attention on a clearly defined and time limited goal.

One of the first major contracts awarded by NASA on 10 August 1961, only 10 weeks after Kennedy's announcement, was for the guidance and navigation computer.[25] This was awarded to MIT's Instrumentation Laboratory, now the Charles Stark Draper Laboratory, and it was the starting point for the Apollo Guidance Computer (AGC).[26] The Instrumentation Laboratory had previously worked on the Polaris missile and had completed a recent study for the Air Force on the feasibility of a mission to fly-by and photograph Mars.[27] While there was experience in MIT on guidance systems, the Apollo Guidance Computer would require significant innovation in both hardware, software and software engineering.

At the time the contract was signed in 1961, computer technology was still in its infancy. The Whirlwind computer was built by MIT in the early 1950s and was designed to operate as an aircraft flight simulator. It was much more powerful than ENIAC and could do 20,000 operations per second.[28] It was retired as obsolete in 1959, having never achieved its goal as a pilot trainer. Over the 1950s, computer circuitry moved from using large, power hungry and unreliable vacuum tubes to more efficient transistors. Computer memory moved from using magnetic drums to using

---

[24] Mieczkowski, *Eisenhower's Sputnik Moment*, 265
[25] John Tylko, "MIT and navigating the path to the moon", *AeroAstro*, 2009, accessed 19 January 2025, https://web.mit.edu/aeroastro/news/magazine/aeroastro6/mit-apollo.html ; Also see "The First Apollo Mission Contract goes to…", Hack The Moon, accessed 8 May 2025, https://wehackthemoon.com/people/first-apollo-mission-contract-goes
[26] Philip D. Hattis, "How Doc Draper Became The Father Of Inertial Guidance", *Advances in the Astronautical Sciences AAS/AIAA Guidance, Navigation and Control* 2018, volume 164. :12
[27] Milton B. Trageser, "A recoverable interplanetary space probe", *Astronautics*, Volume 5, Issue 5, May 1960: 32-35. There is also a good description of early ideas of an inertial guidance computer that influenced the Apollo Guidance Computer at R. Alonso and J.H. Lanning, Jr., "R-276, Design Principles for a General Control Computer", MIT Instrumentation Laboratory, April 1960.
[28] Hall, *Journey to the moon*, 31

core memory. Speed and efficiency increased over the decade. Interacting with a computer was not real time, in that they were programmed via switches or paper tape or punch cards, and you received the output at some future point via a printer. Commercial development and use of computers grew during the 1950s, starting with the UNIVAC computer in 1951.[29] By the time of Kennedy's moonshot speech, the state of the art for general purpose commercial use was the IBM 1401. This computer was initially shipped in 1959, weighed 4,000 kg, consumed 13,000 watts of power, cost $6,500 / month to rent, contained 10,600 transistors, had a clock speed of 87 KHz, and was programmed via punch cards with program output via a printer.[30]

Software Engineering did not exist at this point, although programming did. Because computers were not very powerful, programming tasks were constrained to relatively simple, and therefore relatively small, tasks that did not require much engineering discipline. The process of programming had evolved from physically rewiring the ENIAC computer, to providing instructions on paper tape that were read by a tape reader, to providing instructions on stacks of punch cards via a card reader. Each section of tape or each punch card represented one computer instruction. The process of programming with punch cards is described well in *Programming with Punched Cards*.[31] In summary, the programmer needed to understand the machine instruction set of the computer they were working on, then the code was written by hand on coding sheets. These sheets were then taken to key punchers. Their job was to convert the coding sheets into a stack of punch cards. The stack was then taken to the computer operator who scheduled a job to ingest the stack, verify its correctness and provide a printout of the program source. The program source could then be reviewed by the programmer. Updates to the program could be made by changing, adding or removing specific punch cards in the stack and resubmitting. When the program was deemed correct, the operator would schedule another job to run the program, and the output was provided in another printout to the programmer. This is the process that the software teams in MIT followed.[32] Up until the Apollo Guidance Computer, the largest software effort was for the SAGE (Semi-Automatic Ground Environment) system, which was an air defence reconnaissance system for the United States deployed in the late 1950s. The system was also built by MIT.

---

[29] Martin Campbell-Kelly, *Computer: A History of the Information Machine*. 3rd ed. Boulder: Westview Press, 2014, 107

[30] Robert Garner and Rick Dill, "The Legendary IBM 1401 Data Processing System." *IEEE Solid State Circuits Magazine* 2, no. 1 (2010): 35.

[31] Dale Fisk, *Programming with Punched Cards*, 2005, accessed 28 February 2025 https://web.archive.org/web/20090325223903/https://www.columbia.edu/acis/history/fisk.pdf

[32] Brock, "Oral History of Margaret Hamilton", 24

The politics of the Space Race had created the need for the Apollo Guidance Computer. The existing computer technology was wholly inadequate for the task, software engineering did not exist, and Margaret Hamilton was working as a programmer, but on a different project.

Margaret Elaine Heafield was born in 1936 in Paoli, Indiana, USA to teachers, Esther and Kenneth Heafield. She went to high school in Hancock, Michigan and worked part time in the Arcadian Mine tourist attraction. She went to college at the University of Michigan at Ann Arbor in 1954, but transferred to Earlham College, Richmond, Indiana in 1955. She majored in Mathematics and minored in Philosophy, graduating in 1958.[33] While at Earlham, she met Jim Hamilton, whom she married in 1958. After moving to Boston in 1959, they agreed that Jim would pursue his post graduate studies first while Margaret supported the family, with the intention that she would go to graduate school when Jim finished. This was also the year that their daughter, Lauren, was born.

Margaret procured a research assistant job in MIT, working in the Meteorology Department under Dr. Edward Lorenz. Lorenz was researching mathematical models for predicting weather. Margaret's job was to convert his mathematical equations into computer code, via assembly language that was then converted into holes in a paper tape, which was then ingested into the LGP-30 computer.[34] She was becoming a programmer, even though that job role had only recently been created. Dr Lorenz is credited with discovering the Butterfly Effect, which is part of Chaos Theory.[35] He acknowledges Margaret Hamilton's help in his 1962 paper, "The writer is greatly indebted to Mrs. Margaret Hamilton for her assistance in performing the many numerical computations which were necessary in this work".[36]

In 1961, Margaret moved to a new job with a higher salary. This role was in MIT's Lincoln Laboratory where she worked on the SAGE project as a programmer on the AN/FSQ-7 computer.[37] In all her roles, Margaret excelled at programming and leadership, and this was to continue throughout her career. She continued to work on SAGE until 1963, at which point she moved back to MIT's meteorology department to be closer to her family. In late 1964, she noticed an advertisement for MIT's Instrumentation Laboratory in the Boston Globe. They were looking for

---

[33] Maurer, *The Woman in the Moon*, 227
[34] Maurer, *The Woman in the Moon*, 73
[35] G. Ambika, "Ed Lorenz: Father of the 'Butterfly Effect'." *Resonance* 20, no. 3 (2015): 198-205.
[36] E. N. Lorenz, "The statistical prediction of solutions of dynamical equations", *Proc. Int. Symp. on Numerical Weather Prediction*, Tokyo, Japan, Meteorological Society of Japan, 1962, 629–634.
[37] Brock, "Oral History of Margaret Hamilton", 14

computer programmers for the Apollo program to "assist in data processing, trajectory analysis, instrument data reduction and coding of flight programs".[38] Although Margaret was, at this time, strongly considering continuing her post graduate degree in Mathematics, she jumped at the chance to apply and was successful. She started working as a computer programmer for the Apollo Guidance Computer at the Instrumentation Laboratory in April 1965.[39]

At the time of the contract between NASA and MIT, the requirements for a guidance and navigation computer for the Moon missions were unclear. The overall approach to Apollo was yet to be agreed. In June 1962, NASA converged on the Lunar Orbit Rendezvous method, where a single launch vehicle puts a smaller spacecraft into a lunar orbit. Part of this spacecraft, the Lunar Module (LM), lands on the Moon and returns to the orbiting spacecraft, the Command and Service Module (CSM), which then returns to the earth. The requirements for the computer then converged on guidance and navigation for all phases of the mission, including descent and ascent from the lunar surface. This included ongoing calculation of the spacecraft position and trajectory, integration with positioning controls such as the inertial management unit, control of the spacecraft's engines and real time management of the astronaut's display and controls. There were design constraints on physical size, weight and power as it had to fit in a small spacecraft with highly constrained resources.

The hardware for the Apollo Guidance Computer went through a series of evolutions. Initially based on MIT's feasibility study for a Mars flyby mission, the architecture for the AGC was based on the Mod-3C computer. The first laboratory-based version, the AGC-3, became available in November 1962.[40] It had a clock speed of 19.7 microseconds, 1K of RAM and 12K of ROM. MIT made the enormous decision to become the first adopter of integrated circuits in the next version, the AGC-4. Integrated circuits would reduce the size and weight of the computer and increase the compute speed, without any reduction in reliability. However, it was a risky decision as integrated circuits had only been invented in 1959 and were relatively unknown. The AGC-4 became available in early 1963.[41] It had a clock speed of 11.7 microseconds, 1K of RAM and 24K of ROM.[42] The first flight prototype, the AGC-4B, was tested in January 1964. The first production

[38] Advertisement in *Boston Globe*, 18 October 1964, 109, accessed 30 June 2025, https://bostonglobe.newspapers.com/image/433781571/
[39] Maurer, *The Woman in the Moon*, 94
[40] Hall, *Journey to the moon*, 73
[41] Hall, *Journey to the moon*, 87
[42] Hall, *Journey to the moon*, 90. The Command Module had 24K ROM. The Lunar Module only had 12K ROM.

computers, entitled Block I AGC, and based on the AGC-4B, were produced by Raytheon in August 1964 and these were used in the initial test and unmanned missions.[43] Compute power and available memory were expanded in a new Block II configuration, while also decreasing physical size, weight and power consumption. It had a clock speed of 11.7 microseconds, 2K of RAM and 36K of ROM. The Block II AGC became available in the spring of 1966.[44] Block II AGCs were used on all manned missions. The final version weighed 32kg, used 55w of power, contained 5,600 integrated circuits and its dimensions were 60cm x 30cm x 15cm.

Compared to contemporary computer technology, it was a technical marvel. It was compact, powerful, interactive and consumed a small amount of power.

The software effort in the Apollo Guidance Computer was massive for the 1960s and is still very large by comparison to software projects today. In late 1961, the team was only ten people. By mid 1965, around the time when Margaret Hamilton joined the team, there were one hundred people. At its peak size in mid 1968, there were approximately four hundred people. As Figure 1 below shows, the hardware effort was front loaded in the early 1960s, but by late 1966, the software effort was taking most of the effort, indicating the importance of the software effort in the second half of the Apollo program.

[43] Hall, *Journey to the moon*, 100
[44] Hall, *Journey to the moon*, 128; Also see Hall, "R-700, Volume III, Computer Subsystem", 56-57 for a more in-depth timeline of Block I and Block II delivery dates.

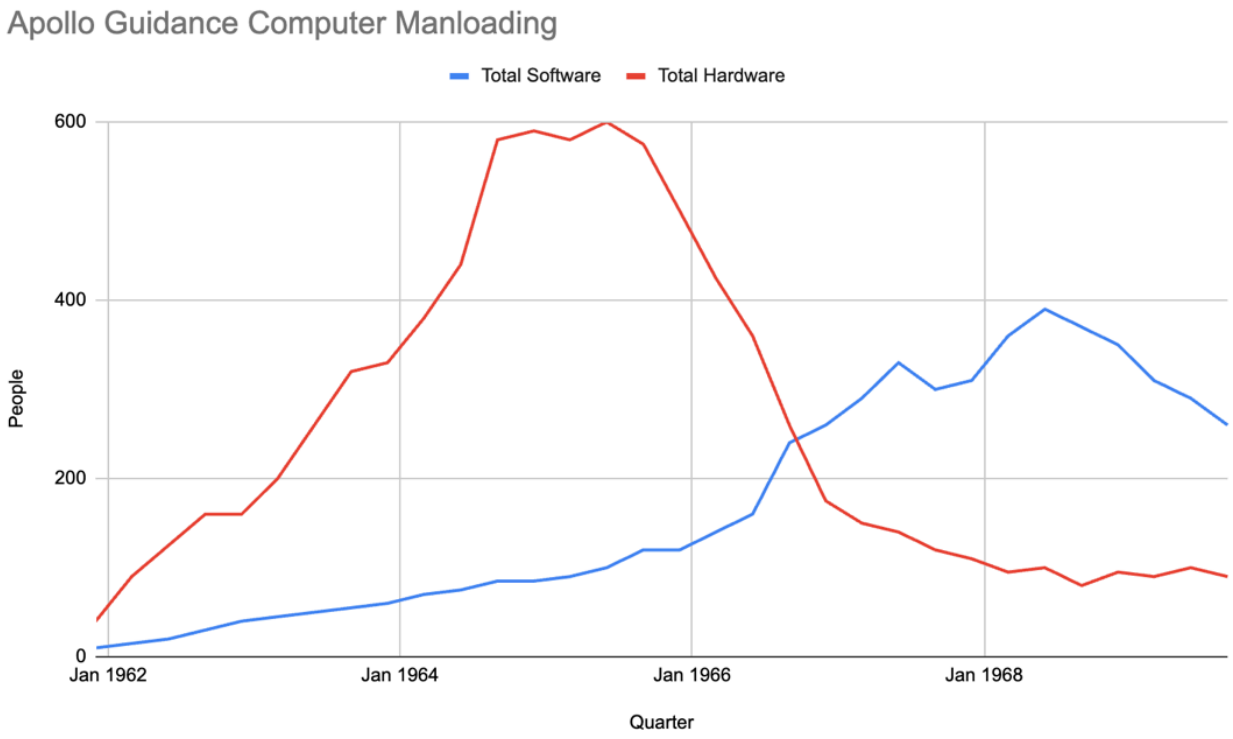

Figure 1 Total people working on AGC software and hardware, measured on a quarterly basis.[45]

For the software teams, they had to write software for both the Block I and Block II computer configurations. Different software was required for the Command and Service Module (CSM) and the Lunar Module (LM). Also, different software was required for unmanned and manned missions. Each had slightly different requirements. Over the life of the project, the software went through several major releases to support different hardware and mission constraints, which is summarized, up to Apollo 11, in Figure 2.

[45] Johnson and Giller, "R-700, Volume V, The Software Effort", 20-21; The graph presented here is based off the 'total hardware' and 'total software' lines in the primary source.

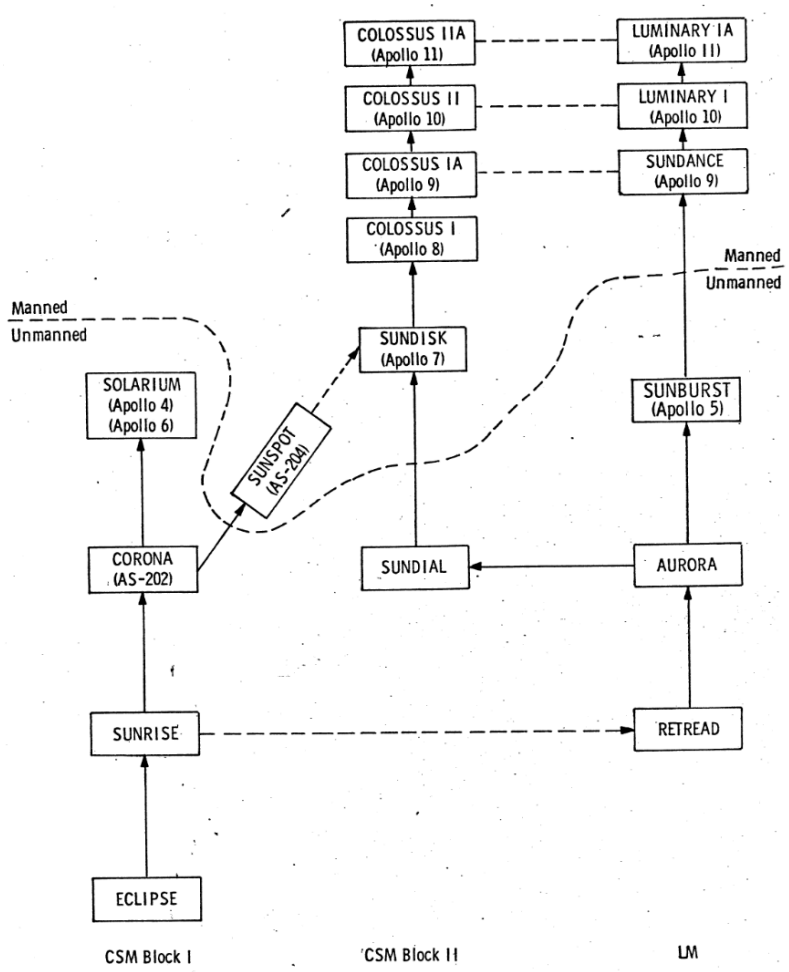

Figure 2 The evolution of the AGC software, across Block 1 & II configurations, CSM and LM, and manned/unmanned missions, up to Apollo 11.[46]

Software Engineering practices, as we understand them today, define the need for a functional requirements document and a design document. The functional requirements document describes what the software should do. The design document describes how the software will be implemented. Both steps are required to get agreement on why the software is needed, what the software will do, what constraints exist on the software and how the software will be implemented.

As the hardware effort was in full swing, the software effort was starting to ramp up in 1962. The first aspect to be coded was the operating system for job management and for expanding the instruction set of the computer. These were implemented without any formal requirements or

[46] Johnson and Giller, "R-700, Vol V, The Software Effort", 7 Also, see the Appendix in this document for a detailed breakdown of software releases for the full Apollo program.

design documents, indicating a lack of Software Engineering practices at this point. These three parts of this early work are named: the Executive, the Waitlist and the Interpreter. The Executive is a priority-based job manager that ensures that the highest priority jobs are being executed first. The Waitlist keeps track of, and executes, short jobs (such as keypresses) on time. The Interpreter expands the computer's basic instruction set in a memory efficient manner by encoding complex trigonometrical and vector equations in basic machine instructions and making these instructions available to the rest of the code.[47]

There is no requirements document or design document for the Executive, Waitlist and Interpreter. This indicates that Software Engineering processes were not present at this point.[48] User training documentation was retrofitted after the fact by the programmers. It is possible to identify when these functions were implemented. We know that they were available in the SUNRISE/33 software release in July 1964.[49] They were also available in the AGC-3 computer in March 1963.[50] However, they are not mentioned in a Mod-3C document from November 1961, implying they were implemented sometime in 1962.[51]

In comparison, on the hardware side, the design and evolution of the physical computer was managed as an engineering exercise. This is a further indication of the minimal understanding of how to build software and the high level of understanding on how to build physical devices. From the beginning of the project, there was a Design Review Board and a Change Control Board. These processes are discussed in a Technical Progress Report from December 1962.[52] Up until the end of 1969, over 37,000 Technical Data Release or Revisions (TDRRs) impacting the hardware design of the computer used these processes.[53] However, the software side of the AGC did not use these

[47] "SINGLE_PRECISION_SUBROUTINES.agc.html", The Virtual AGC Project GitHub Repository, accessed 20 January 2025, https://www.ibiblio.org/apollo/listings/Comanche055/SINGLE_PRECISION_SUBROUTINES.agc.html ; For example, this is the implementation of Sine and Cosine in the Interpreter.
[48] Eldon C. Hall, "R-410 General Design Characteristics of the Apollo Computer", MIT Instrumentation Laboratory, May 1963, 7 has a reference to what might be a requirements document for the Interpreter at Muntz, Charles A, "MIT/IL AGC Memo #2, A List Processing Interpreter for AGC 4", 7 January 1962. However, this document is not available.
[49] R. Battin et al "R-467, The Compleat Sunrise, Being a Description of Program SUNRISE, (SUNRISE 33 – NASA DWG #1021102)", MIT, September 1964, 5-47
[50] Albert Hopkins, Ramon Alonso and Hugh Blair Smith, "R-393, Logical Description for the Apollo Guidance Computer (AGC 4)", 5 March 1963, page 2-18
[51] R. Alonso, J.H. Laning, Jr. and H. Blair Smith, "E-1077, Preliminary Mod 3C Programmers Manual", MIT Instrumentation Laboratory, November 1961. Nor are they mentioned in R. Alonso and J.H. Lanning, Jr., "R-276, Design Principles for a General Control Computer", MIT Instrumentation Laboratory, April 1960.
[52] Anonymous, "Report E-1236. Monthly Technical Progress Report. Project Apollo Guidance and Navigation Program. Period Jun 11, 1962 through July 17, 1962", MIT Instrumentation Laboratory, 17 July 1962, 1-6
[53] Hand, "R-700, Volume I, Project Management", 24-25

processes. Software, or programming, was not considered an engineering exercise, as after all, there was nothing you could touch or see in the software, it was merely programming the hardware to behave in a certain way. As we shall see, this was to become very problematic as the 1960s progressed and the amount of effort on the software side increased significantly.

In another indication of the lack of understanding of the importance of software, the organisational chart for MIT's AGC work from May 1962 does not include a software group. The closest groups are the Computer Division, run by E.C. Hall and the Space Guidance Analysis Division, run by R.H. Battin. Hall was one of the key people that built out the hardware design. Battin, in later years, would eventually be one of the key people on the software side.[54]

The first release of the AGC code was entitled ECLIPSE, and it was released in February 1964.[55] This was a test release that brought together the fundamental parts of the software: Executive, Waitlist, Interpreter and the programs for interacting with the Display and Keyboard (DSKY) in the Block 1 hardware configuration. It was used to test the manufacturing process for the fixed memory, which was a woven rope core memory.[56] Although we know of its existence from the historical record, there is no documentation outlining the requirements, design or test plans for ECLIPSE, so there is no indication of Software Engineering practices being used at this point. The closest we get to a functional description is *MIT Instrumentation Laboratory, DG Memo No.88, Description and Status of AGC Programs* from January 1964 which hints at problems in the development process:

> "Several 'lists' of computer programs have been written, and no two lists call the programs by the same names or strike the division between successive programs at the same point in the mission, even when written by the same author. Increased effort has been expended recently to produce a program organization".[57]

---

[54] Anonymous, "Apollo Organization", MIT Instrumentation Laboratory, May 1962, accessed 3 July 2025, https://www.ibiblio.org/apollo/Documents/ApolloOrg-1962-05.jpg
[55] Hall, "R-700 Vol III, Computer Subsystem", 152
[56] Hall, "R-700 Vol III, Computer Subsystem", 155
[57] J. L. Nevins, "MIT Instrumentation Laboratory, DG Memo No.88, Description and Status of AGC Program", 20 January 1964, 1

Fred Martin, who, later in the timeline of the program, was the MIT project manager for the Command and Service Module software deliverables and liaised closely with NASA, remembered this early development phase as follows:

> "There was the small group, seat-of-the-pants effort that was unfettered and unhindered, for the most part, by NASA, and devoid, pretty much, of real bureaucracy. And I think that, as Dan [Lickly] pointed out, you had-- a lot of the fundamental engineering work was done during that period. And it was very fortunate that we were able to do it unfettered. Because I think that a lot of the fundamental issues on how to get to the Moon, let's say, were solved in this period of isolation-- almost isolation, and small groups".[58]

This quote is quite illuminating, as it points to the lack of oversight from NASA during this process and it equates Software Engineering processes with bureaucracy. Even today, Software Engineering processes are sometimes viewed by software developers as bureaucracy as they do impose a structured methodology on the software development process.

One of the areas that we can begin to see the development of Software Engineering is in the testing process of the AGC, which verifies that the software functionality matches the requirements. Modern Software Engineering requires code to be testable and for a project to have a testing strategy with different levels of testing: unit testing, functional testing, integration testing and performance testing. Also, releases are gated on passing an agreed level of testing.

MIT started to achieve this, initially more through necessity than foresight. As ECLIPSE was being developed between 1962 and 1964, MIT had a problem in that neither the physical AGC computer nor any of the spacecraft instruments that the computer interacted with were available yet. Therefore, it was impossible to do any testing of the software. The approach taken to mitigate this issue was to simulate everything based on the best understood specifications of the yet-to-be-built computer. This 'All Digital' simulator, which itself was a significant software effort, ran on IBM and Honeywell mainframes in a laboratory at MIT. It consisted of two main components: the 'AGC Instruction Simulator' which implemented the instruction set of the physical computer, along with simulated fixed and erasable memory and the 'Environment' which simulated the spacecraft instruments, spacecraft dynamics, trajectory changes, celestial body movements and gravitational

[58] MIT Video Productions, "Apollo Guidance Computer Project – MIT History Conference pt.2 – 2001", 14 September 2001, 49:45 – 50:39, https://www.youtube.com/watch?v=quIfco4RLCg

impacts.[59] It had numerous options for stopping, starting, recording, replaying, tracing and dumping the environment during the tests. This 'All Digital' simulator became operational in September 1962 and was used to test ECLIPSE and all future releases.[60] In fact, it was also vice versa, as the ECLIPSE software was also used to uncover bugs in the simulator. Another additional advantage of the simulator was that it could be constantly updated as the known details for the Apollo Guidance Computer and the spacecraft were being changed. The 'All Digital' simulator was the testing work horse for MIT and played the largest part in the testing and verification. Although there does not seem to be an early design document for the simulator itself, it is documented in significant detail, although much later in the Apollo program, in *Digital Simulation Manual*.[61]

In summary, by early 1964, there were approximately sixty people working on the software for the Apollo Guidance Computer, as compared to almost six hundred people working on the hardware. The first parts of the software had been implemented and the first release, ECLIPSE, had been released. The 'All Digital' simulator was available. All of this had been done without any Software Engineering practices.

[59] Johnson and Giller, "R700, Vol V, The Software Effort", 86
[60] Hall, *Journey to the moon*, 157
[61] Pieter Mimno, "R-599 Digital Simulation Manual", MIT Instrumentation Laboratory, January 1968. A later document goes into more detail – F.K. Glick and S.R. Femino, "E-2475 A Comprehensive Digital Simulation for the Verification of Apollo Flight Software", MIT Charles Stark Draper Laboratory, January 1970

2 Initial Inroads into Software Engineering (1964–1966)

The next phase in the development of the Apollo Guidance Software shows improvements in documentation and testing at MIT and shows how NASA made initial approaches on how to manage large software projects, although these were not adopted by MIT.

The next major release of the AGC software was the SUNRISE release in July 1964. It is described in detail in a document produced by MIT and Raytheon in March 1964.[62] This is an improvement on the previous approach as it was produced before the actual release of the software. While this is a technical training document, it does contain a relatively complete functional description of the AGC hardware and the software that runs on it. Expanding on what is the ECLIPSE release, SUNRISE also includes Mission functions such as navigation and Auxiliary functions such as displaying information.[63] SUNRISE was used to test the initial AGC hardware. Another detailed end-user document produced in September 1964, after the release date, describes how to use the Display and Keyboard DSKY to control the computer. It should be noted that this document includes the first mention of the 1201 and 1202 program alarms that happened during the Apollo 11 landing.[64] Documentation is improving at this point, but there is still no documented design or test plans or anything that resembles a user acceptance plan. In today's understanding of Software Engineering, these are all required documents.

From the early 1960s, NASA understood there was an issue with managing large scale software projects, including the AGC. Working with Bellcomm, the concerns and initial proposals for managing these concerns are contained in a November 1964 document *Management Procedures in Computer Programming for Apollo – Interim Report*.[65] The report asserts that "computer programming is a describable, orderly process having a determinable, predictable end product".[66] It

---

[62] Anonymous, "Apollo Guidance Computer Information Series, Issue 15, Block I Apollo Guidance Computer Subsystem, FR-2-115", 27 March 1964. The Apollo Guidance Computer Information Series (AGCIS) was published between 1963 and 1966 to inform the technical staff of MIT and Raytheon about the Apollo Guidance Computer subsystems.
[63] Anonymous, "FR-2-115", page 15-63.
[64] Battin et al., "R-467, The Compleat Sunrise", 79
[65] W.M. Keese et al., "Management Procedures In Computer Programming Apollo – Interim Report", Bellcomm Inc, 30 November 1964
[66] Keese et al., "Interim Report", abstract

describes the problems of lack of definitive requirements, complex interfaces, lack of quality, lack of programming experience and lack of understanding of costs. It describes computer programming as "neither mysterious nor 'artistic'. Rather it is describable, orderly, and predictable and therefore susceptible to close management control".[67] The document then details potential solutions for further study, specifically defining controls for "proper documentation; adherence to program standards in the area of coding, flow charting, program verification; costs, schedules and requirements; methods of payment; and acceptance testing".[68] However, at this time, none of these recommendations were used by MIT's Apollo Guidance Computer software development team.

The first Apollo launch that included an AGC was the AS-202 mission on 25 August 1966. This was a ballistic suborbital unmanned mission with the goal of testing the main components of Apollo, including the Saturn IV B rocket and the guidance systems. This flight included the Command and Service Module but not the Lunar Module. The post launch report indicated that "The guidance and navigation subsystem … demonstrated their adequacy for a manned orbital mission".[69] The software release for this launch was called CORONA and it was released for manufacturing in January 1966.[70] In terms of documentation, CORONA is the first time we see the *Guidance and Navigation System Operations Plan* (GSOP).[71] The Guidance System Operations Plan evolved during the development of the AGC to become the formal specification and NASA's signoff document for MIT's software efforts for each launch. It evolved through a series of discussions among the personnel at NASA and MIT's Instrumentation Laboratory.[72] Each subsequent delivery from MIT for a manned or unmanned flight included a GSOP and it grew from a relatively compact one volume document for AS-202 to a six-volume document in later missions. For the AS-202 mission, the GSOP details the Guidance & Navigation Flight Operations, Guidance Equations, Control Data, Error Analysis, Configuration, Instrumentation and Performance Analysis. The document was initially written in January 1965, one year ahead of the software release, and was revised in July 1965 and again in October 1965 to take into account feedback from NASA. It is an incredibly detailed definition of all aspects of the AGC, including

[67] Keese et al., "Interim Report", section 3.1.1
[68] Keese et al., "Interim Report", section 4.0
[69] Anonymous, "MSC-A-R-66-5, Postlaunch Report for Mission AS-202 (Apollo Spacecraft 011)", NASA Manned Spacecraft Center, 12 October 1966. section 11.0
[70] John E. Miller and Ain Laats, "E-2397, Apollo Guidance and Control System Flight Experience", MIT Instrumentation Laboratories, June 1969, 1
[71] John M. Dahlen et al., "R-477 Guidance and Navigation System Operations Plan, Apollo Mission 202", January 1965
[72] Johnson and Giller, "R-700, Vol V, The Software Effort", 15

detailed handwritten flow charts of all the actions that the AGC takes during the mission, for example the actions at the time of launch.[73] The presence of an acceptance plan shows the maturing of the Software Engineering processes.

The CORONA release and the AS-202 mission also exercised the next evolution of testing infrastructure. While the 'All Digital' simulator remained the workhorse of testing, it had some limitations. It did not use the physical Apollo Guidance Computer hardware, it was not interactive, so it could not test the astronauts interacting with the computer via the DSKY and the mathematical models of the Environment were not sufficiently accurate. What was needed was an integration testing tool that could verify that the Apollo Guidance Computer programs matched expected requirements in an environment that, as much as possible, matched a working spacecraft. The solution to these limitations was the 'Hybrid' simulator, which used a combination of analogue and digital computers plus the physical AGC computer, along with guidance instruments to accomplish real time, interactive simulation of flights.[74] For flexibility, the fixed memory within the AGC was replaced by a core rope simulator, which enabled different versions of the software to be easily loaded. The user (a designer, a programmer, a human-factors engineer or an astronaut) interacted with the simulation via the DSKY, hand controllers and switches on a mock-up of the spacecraft. To better understand the complexity, and hence utility of this simulator, we will go into a little more detail. The simulator consisted of an analogue simulation of the mathematical model of the spacecraft dynamics (such as mass changes, centre of gravity changes and gravitational effects), a simulation of the inertial subsystem (so the spacecraft knows its attitude and velocity), a simulation of the optical subsystem (so the spacecraft knows its position and orientation in space), a Command Module mock-up, a Lunar Lander mock-up, and the core rope simulator attached to the physical AGC hardware.[75] Across this whole system, specific scenarios could be tested, and the behaviour of the computer programs could be analysed dynamically or post the test.[76] The 'Hybrid' simulator was first used for the AS-202 unmanned mission and was expanded upon for future missions. It was used to test all future missions. The importance of the 'Hybrid' simulator is evident from a memo from October 1966 – "It was discovered during program development for

[73] Dahlen et al., "Apollo Mission 202", 3-12
[74] Johnson and Giller, "R-700, Vol V, The Software Effort", 89
[75] Madeline M. Sullivan, "Hybrid Simulation of the Apollo Guidance and Navigation System", *Simulation*, Vol 7 No 1 July 1966: 28-32
[76] Philip G. Felleman, "E-2066 Hybrid Simulation of the Apollo Guidance Navigation and Control System", MIT Instrumentation Laboratory, December 1966, 1

AS-204/205 [the planned first manned mission] that the hybrid facility at MIT was an extremely valuable tool for program debugging".[77]

The next significant step for the AGC (and more broadly, for the Apollo program) was a manned mission in late 1966 or early 1967. This was the target of the SUNSPOT release. However, by this point, the complexity of the software and the hardware had significantly increased. There was also intense pressure from NASA to meet deadlines. The project organization and processes had been intentionally kept informal, as evidenced by the limited documentation efforts of previous releases. However as of late 1965, around the time of the CORONA release, "the development of the computer programs became the central problem of the entire Apollo project at the Instrumentation Laboratory".[78] There was a shortage of experts in MIT and documentation fell behind because "These engineers were so busy developing the programs that they had little or no opportunity to document them".[79]

Margaret Hamilton took some of the learnings from her work on SAGE into the Apollo Guidance Computer software development, such as putting comments in code.[80] Common coding standards and coding style guidelines are a well understood aspect of Software Engineering as it improves the understandability and maintainability of the code. There is no documented evidence of a formal coding style in the Apollo Guidance Computer, apart from code commenting.

In summary, towards the end of 1966, there were approximately two hundred and fifty people working on the software for the Apollo Guidance Computer. The computer and its software had a successful maiden flight in the unmanned AS-202 mission, the Hybrid simulator had been developed, and we see the emergence of the first Guidance and Navigation Systems Operations Plan documentation. These were all good signs of an emerging Software Engineering mindset. However, the recommended process improvements from Belcomm had not been implemented and the existing informal process was straining under the pressure to deliver. This was soon to become a crisis.

---

[77] Howard W. Tindall, "66-FM1-124. Program Development Plans are coming!!", 12 October 1966, accessed on 21 March 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf
[78] Hand, "R-700, Volume I, Project Management", 37
[79] Hand, "R-700, Volume I, Project Management", 37
[80] Brock, "Oral History of Margaret Hamilton", 15

3 Crisis and the Emergence of Software Engineering (1966–1968)

1966 was a crunch time for the development of the AGC that ultimately resulted in a better Software Engineering process for the rest of the Apollo program. The software effort was becoming the critical deliverable holding up the whole effort to land on the Moon. The tragic Apollo 1 fire in January 1967 gave NASA and MIT the time and space to re-examine how the effort was being managed and to introduce Software Engineering practices that paid dividends in future years.

After the CORONA software was released in January 1966, the team focused on the SUNSPOT release, which was for the first manned mission, originally scheduled for late 1966. There were two launches that were being targeted: the AS-204A launch, which was a manned 14-day earth orbit mission and the AS-204B, which had the same objectives, but would be unmanned had the launch vehicle not been qualified for manned flights.[81] The objective was to test the functions of the Command Module in earth orbit. The main additional requirements on SUNSPOT revolved around more elaborate astronaut-interface display programs. As this was to be the first manned mission, program interaction with the astronauts was a key feature.[82] The main impediment to the release of SUNSPOT was overuse of the computer memory. Basically, the software had outgrown the physical memory constraints of the hardware. As Howard "Bill" Tindall, Director of Flight Operations at NASA, noted:

> "Our basic problem seems to center on the time available to prepare the computer programs for these flights and on the fact that the computer is not big enough to contain all of the programs which appear to be either required or highly desirable for the mission".[83]

---

[81] Courtney G. Brooks and James M. Grimwood, Loyd S. Swenson, "Preparations for the First Manned Apollo Mission" in *Chariots for Apollo: A History of Manned Lunar Spacecraft*, NASA Special Publication-4205, NASA History Series, 1979. Accessed 10 February 2025, https://solarviews.com/history/SP-4205/ch8-7.html

[82] Johnson and Giller, "R-700, Vol V, The Software Effort", 9

[83] Howard W. Tindall, "66-FM1-59. Spacecraft computer program requirements for AS-207/208, AS-503, and AS-504", 12 May 1966, accessed on 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

These, along with other issues impacting the schedule were resolved via several "Black Friday" meetings where hard decisions were made about what to include or not include in the software release. According to MIT,

> "These meetings became emotional because of disagreement about what was, in fact, nonessential. Nonetheless, difficult compromises resulted in the current fixed-storage capacity being reduced sufficiently to allow inclusion of every essential routine".[84]

Other memos from Howard Tindall describe the difficult situation that year: concern about getting a detailed schedule (entitled a Program Development Plan), concerns about the amount of manual work required by the astronauts to interact with the computer, reviews of the developing AS-207/208 GSOP document, concerns about the testing capacity and concerns on the availability of new testing hardware and its impact on the flight schedule.[85] With multiple software releases being worked on concurrently and significant schedule pressure, the lack of Software Engineering process gives the strong impression that the overall project was lurching from one chaotic situation to the next.

This was becoming obvious to most parties, including MIT.

> "An informal organization was workable in the five earliest, innovative years of Apollo. Thereafter, this approach exhibited shortcomings related to problems of communication, documentation, and the changing external environment as the project grew larger and the type of work changed significantly. A more formal structural approach was viable".[86]

It was around mid 1966 that Margaret Hamilton first started using the term "Software Engineering" for the work that she and the teams around her were doing.[87] At this time, she was a programmer on the Command and Service Module code and had not yet assumed a formal leadership role. From an oral history of Hamilton in 2017, "I said, 'Why don't we call this one the hardware

---

[84] Johnson and Giller, "R-700, Vol V, The Software Effort", 15
[85] Howard W. Tindall, "66-FM1-70. Spacecraft computer program status report", 2 June 1966; Howard W. Tindall, "66-FM1-100. Notes regarding the AS-207/208 Guidance Systems Operation Plan (GSOP) meeting with MIT", 30 August 1966; Howard W. Tindall, "66-FM1-170. More interesting things about our work with MIT", 28 November 1966; Howard W. Tindall, "66-FM1-191. MIT's digital computers are saturated until the IBM 360 becomes operational", 22 December 1966, accessed 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf
[86] Hand, "R-700, Volume I, Project Management", 41
[87] Maurer, *The Woman in the Moon*, 155

engineering part and this the software engineering part?' It was like I just came up with a term to say let's name this part in this whole system definition".[88] And from an interview with Hamilton in 2014, "I fought to legitimize software so that both software engineering and those who built it would receive the respect they deserved, so I began using the term 'software engineering' to differentiate it from hardware and other forms of engineering".[89] This is an important event as others in MIT started to appreciate the need to bring engineering discipline to the software development process.

NASA continued to understand there was an ongoing oversight problem with the development of software in Apollo. In September 1966, almost two years after the previous document *Management Procedures in Computer Programming for Apollo – Interim Report* was published, the concepts are expanded upon in another document from Belcomm, written by some of the same authors, entitled *Procedures for Management Control of Computer Programming in Apollo*.[90] While acknowledging that "The commitment to use computers in Apollo came at a time when formal and proven management techniques for the control of computer program production were either unknown or undocumented", it goes on to describe specific deliverables such as System Requirements, Detailed Design, Test Specifications and specific milestones along the definition, acquisition and operation phase of software delivery.[91]

The Apollo 1, formerly known as AS-204A, fire happened on 27 January 1967 during a ground test of the Command Module. It resulted in the deaths of three astronauts: Roger Chaffee, Gus Grissom and Ed White. The tragedy affected all parts of the Apollo program and forced a re-examination of all functions within Apollo to understand what happened and to ensure nothing equivalent could happen again. Although the AGC was not culpable for any part of the tragedy, the delay to the overall program gave NASA and MIT time to reset, ensure proper processes were in place and to retarget their efforts for future missions. The next manned mission would not be until October 1968.

---

[88] Brock, "Oral History of Margaret Hamilton", 33
[89] James Rubio Hancock, "Margaret Hamilton, the programming pioneer who took Apollo to the Moon", El Pais, 25 December 2014, accessed 16 January 2025, https://verne.elpais.com/verne/2014/12/11/articulo/1418314336_993353.html
[90] B.H. Liebowitz and C.S. Sheppard, E.B Parker III, "Procedures for Management Control of Computer Programming in Apollo", Bellcomm Inc, 28 September 1966
[91] Liebowitz and Sheppard, Parker, "Control of Computer Programming", 1

The last memo from Howard Tindall before the Apollo 1 fire details an MIT meeting from 26 January 1967 related to the future scheduled AS-206 mission.[92] Interestingly, he says "Obviously our toughest job is going to be wrenching this program out of MIT's grasp, since to them quality still comes before schedule. But that's just a little game we are playing, and I don't consider it unhealthy".[93] This approach would change after the fire. There is a gap of twenty-six days in the memos, possibly because of the fire. The next memo details another hole in the Software Engineering process, which is cost accountability – "We requested MIT to propose at our next meeting how they intend to provide cost accounting for their work in the software area, something which is sadly inadequate at this time".[94]

As 1967 progressed, it was becoming clearer that there were continued serious concerns with the Apollo Guidance Computer. In March, concerns were raised "regarding the adequacy and formality of the final verification testing of the Apollo spacecraft computer programs and its interfaces with other spacecraft systems".[95] Particularly concerning was "that documentation of the test plans and test results was not even adequate to really understand what had been done".[96] In summary, while there was now a specification of what was required, there was a lack of process around ensuring that what was delivered matched the requirements. As part of mitigating these concerns, the outcomes of the 'All Digital' simulation test runs were to become a formal part of the verification and acceptance process.[97] This would eventually turn into a specified level of testing that each release was to attain.

By May of 1967, the pressures associated with the schedule, the testing, the cost accounting, the constraints imposed by the computer hardware, all in conjunction with the impact of the Apollo 1 fire all came to a head. In a Howard Tindall memo entitled *A new spacecraft computer*

---

[92] AS-206 was to be the first mission to launch with a Lunar Module. It was scheduled for April 1967 but was cancelled after the Apollo 1 fire.
[93] Howard W. Tindall, "67-FM1-17. AS-206 Spacecraft Computer Program Newsletter", 31 January 1967, accessed on 17 March 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf . In the margin of this memo, in handwriting, is "Fire about now!"
[94] Howard W. Tindall, "67-FM1-18. Spacecraft Computer Program Development Newsletter", 27 February 1967, accessed on 17 March 2025, https://www.ibiblio.org/apollo/Documents/1967_tindallgrams.pdf .
[95] Howard W. Tindall, "67-FM1-23. Summary of what needs to be done to develop flight confidence in the spacecraft computer programs", 23 March 1967, accessed on 17 March 2025, https://www.ibiblio.org/apollo/Documents/1967_tindallgrams.pdf , 1
[96] Tindall, "67-FM1-23", 1
[97] Tindall, "67-FM1-23", 2. See bullet 5 "It is intended that the so-called Bit-by-Bit simulation facility at MSC play a formal part in the program verification process". The All Digital simulation was also known as the Bit-by-Bit simulation.

development program development working philosophy is taking shape, Tindall describes the move from delivery to a tight official flight schedule to delivering to a higher quality bar, even if this means delivering a little later. “These things are most clearly evident right now on LM-1 [Apollo 5] where it’s almost unthinkable to fly with any known deficiencies in the program”.[98] The final paragraph summarizes the change in approach from just before the Apollo 1 fire, “It should certainly be easier to handle than our previous ‘schedule is king – anything is better than nothing’ type of problem”.[99]

It was also around mid 1967 that a new organisational structure was put in place at MIT to better meet the needs of the growing requirements on software development. Within the Mission Development Division under R.H. Battin, there was now a Guidance Programs group, with the moniker 23B, that had responsibility for the onboard software.[100] In May 1967, 23B was led by E. M. Copps.[101] Within 23B, there was a CSM Group and an LM group, with responsibility for the Command and Service Module software and the Lunar Module software respectively. Margaret Hamilton led the CSM Group in late 1967.[102]

The outcome of discussions between NASA and MIT resulted in the earliest formally documented definition for how the AGC software was to be developed, tested and accepted. This is the *Apollo Guidance Software Development and Verification Plan* written by NASA in October 1967, with input from MIT.[103] It is obviously heavily influenced by the September 1966 Bellcomm document, as the names of several milestones are repeated in both documents.

---

[98] Howard W. Tindall, “67-FM1-39. A new spacecraft computer program development working philosophy is taking shape”, 17 May 1967, 1, accessed on 13 January 2025, https://web.mit.edu/digitalapollo/Documents/Chapter7/tindallgrams.pdf

[99] Tindall, “67-FM1-39”, 2

[100] Hand, “R-700, Volume I, Project Management”, 37-38

[101] Anonymous, “Apollo Organization”, MIT Instrumentation Laboratory, May 1967, accessed 3 July 2025, https://wehackthemoon.com/sites/default/files/styles/hero_extra_large_1x_2000x_/public/media/photograph/45838.jpg?itok=ECT1UPEe

[102] Anonymous, “Apollo Organization”, MIT Instrumentation Laboratory, 30 Sep 1968, accessed 12 October 2025, https://www.ibiblio.org/apollo/Documents/ApolloOrg-Group-23B-only-1968-09.jpg indicates that Margaret Hamilton is leading the CSM Programming group in September 1968. K.W. Greene, "MIT Management Development Plan Meeting #4", MIT Instrumentation Laboratory, 7 Nov 1967, 6, accessed 12 October 2025, https://www.ibiblio.org/apollo/Documents/mit_dev_plan_meeting_04.pdf is the first of twenty weekly management meetings (starting on 13 Oct 1967) that cc’d Margaret Hamilton. As these meetings included the leader of 23B and cc’d the subgroup leaders, it is reasonable to assume the Margaret Hamilton led the CSM group in November 1967.

[103] The Guidance Software Validation Committee, "Apollo Guidance Software Development and Verification Plan”, 4 October 1967

The document defines several key milestones that eventually result in acceptance of the software by NASA. The first milestone is the *Critical Design Review (CDR)*, which approves the preliminary Software Design Specification and precedes the actual development phase.[104] The second milestone is the *First Article Configuration Inspection (FACI)* which reviews and accepts the test plans.[105] This milestone precedes the software qualification and verification phase. The third milestone is the *Customer Acceptance Readiness Review (CARR)*, where the software is formally approved and accepted after passing the qualification and verification phase.[106] At this point, the software is released for memory fabrication, after which, verification and system testing can take place. In parallel, any final updates to the specifications and flow diagrams are completed. The final milestone is the *Flight Readiness Review (FRR)*, where approval to use the fabricated software on a particular flight is granted.[107] The document goes into significant detail on the testing requirements, including the use of simulators and astronaut procedure testing. It also details the software change procedure for changes after the Critical Design Review milestone.

MIT was not enthusiastic with some of these processes and viewed them as an overly bureaucratic imposition from NASA. Hamilton noted "we evolved our 'software engineering' rules with each new relevant discovery while top management rules from NASA went from complete freedom to bureaucratic overkill".[108] Fred Martin, MIT project manager for the Command and Service Module software deliverables, remembered this imposition starting around the Autumn of 1966:

> "And these groups now were fairly large, and NASA wanted to see formal schedules, and time to complete this module, and time to complete that module, and reporting on a monthly-- I can't even remember the schedule, what it was. But things started to build up in reviews and in paperwork, and in monitoring and so on. And I would say that that became a different era of doing work at the lab, and much less fun for a lot of people".[109]

While all this was going on, MIT continued to work on and deliver the next version of the Command Module software, effectively in parallel with the SUNSPOT release. This was entitled SOLARIUM and it was released for manufacture in November 1966. It was used in the unmanned

[104] The Guidance Software Validation Committee, " Development and Verification Plan", section 7.2.2
[105] The Guidance Software Validation Committee, " Development and Verification Plan", section 7.3.2
[106] The Guidance Software Validation Committee, " Development and Verification Plan", section 7.4
[107] The Guidance Software Validation Committee, " Development and Verification Plan", section 7.5
[108] Margaret Hamilton, "The Language as a Software Engineer", ICSE 2018 Plenary Sessions, 31 May 2018, 25:26, https://www.youtube.com/watch?v=ZbVOF0Uk5lU
[109] MIT Video Productions, "Apollo Guidance Computer Project", 53:06 – 53:44

AS-501/Apollo 4 (launch date November 1967) and AS-502/Apollo 6 (launch date April 1968) missions. Apollo 4 was the first mission after the Apollo 1 fire and was the first to test the Saturn V launch vehicle. As per previous releases, its specification was defined in a *Guidance and Navigation System Operations Plan*, and similarly, this was approved many months (July 1966) before the release of the software (November 1966).[110] The interim 'All Digital' simulation results were published in October 1966.[111] While similar to the CORONA release, SOLARIUM replaced the elliptical trajectory supported in CORONA, with support for parabolic and hyperbolic trajectories which matched the return trajectories from a mission to the Moon.[112] Even though this release was before the published date of the *Apollo Guidance Software Development and Verification Plan*, it was of high quality as evidenced in the *Apollo 4 Mission Report* - "All sequencing and computational operations performance by the Apollo guidance computer have been verified to have been correct".[113]

In parallel to SOLARIUM, MIT were also working on the software for the first unmanned Lunar Module mission, entitled SUNBURST. Although running on the same physical computer hardware, this software was very different to the Command and Service module in that it included control of the Lunar Module guidance systems. The software was delivered in October 1967 and flew on Apollo 5 in January 1968. The software was critical to the testing of the Lunar Module descent and ascent stages during this unmanned flight. The seven section GSOP for this mission was first written in January 1967 and so follows the developing pattern of writing the GSOP a long away ahead of the software release.[114]

---

[110] Anonymous, "R-537 Guidance and Navigation System Operations Plan Apollo Mission 501", MIT Instrumentation Laboratory, July 1966

[111] Jay Sampson, "Flight 501 Memo #10, Summary of Results of AS-501 Digital Simulations", MIT Instrumentation Laboratory, 26 October 1966

[112] Johnson and Giller, "R-700, Vol V, The Software Effort", 9. For a detailed list of changes between AS 202 / CORONA and AS 501 / SOLARIUM, see J.A. Simpson, "Flight 501 Memo #8. Programming Changes from AS-202 to AS-501", MIT Instrumentation Laboratory, 7 October 1968

[113] Apollo 4 Mission Evaluation Team, "Apollo 4 Mission Report", NASA Manned Spacecraft Center, Houston, Texas, 7 January 1968, 1-3. Also see R.C. Millard, "MIT Management Development Plan Meeting #6", 21 November 1967, 1 which includes "It was stated by C. Kraft of NASA/MSC that the performance of the AS-501 Mission was perfect and that MIT was to be congratulated."

[114] Ronald S. Burkey, "Apollo Program MIT technical documents from JSC", accessed 6 May 2025, https://www.ibiblio.org/apollo/NARA-SW/IndexOfNaraBox209G.html The January 1967 date comes from the table Box 36, which lists R-527. Also see Anonymous, "R-527 (Rev 2) Guidance System Operations Plan for Unmanned LM Earth Orbital Missions using Program SUNBURST, Section 7 G&N Error Analyses", MIT Instrumentation Laboratory, December 1967

As a gauge of how busy MIT were with the Apollo program at this time, on 21 November 1967, they were working concurrently on six separate releases of the software: SUNBURST (LM software for Apollo 5), SOLARIUM (CSM software for Apollo 6), SUNDISK (CSM software for Apollo 7), COLOSSUS (CSM software for Apollo 8), SUNDANCE (LM software for Apollo 9) and LUMINARY (LM software for Apollo 10).[115] There were approximately three hundred people working on the project at this point. Tracking all of these was becoming a major task itself.

Software Engineering processes continued to develop based on the rapid rate of software development at MIT. The *Apollo Guidance Software Development and Verification Plan* was superseded in 1968 by a triumvirate of documents, one of which is the *Apollo CMC/LGC Software Development Plan.*[116] While keeping the same milestones, this document revised the details of the three milestones up to and including the *Customer Acceptance Readiness Review* milestone. The main difference is the replacement of the System Design Document (SDS) with the Guidance System Operations Plan (GSOP), which had already existed and had the same function as the SDS. The development plan is broken into three phases: Program Definition, Program Implementation and Qualification. Formal transition from the Program Definition to Program Implementation phase is via the *Critical Design Review (CDR)* milestone. Formal transition from the Program Implementation to Qualification phase is via the *First Article Configuration Inspection (FACI)* milestone. Completion of the Qualification phase is via the *Customer Acceptance Readiness Review (CARR)* milestone.

The document also details levels of testing to describe different types of testing. Level 1 testing is low level engineering testing as the software is developed. Level 2 testing are more formal tests than Level 1 that are run on the 'All Digital' simulator. Level 3 testing tests combinations of programs "at the level of astronaut callable programs" in both the "All Digital" and "Hybrid" simulators. Level 4 testing is mission sequencing testing across multiple astronaut callable programs. Level 4 testing is required for the software to move to the Qualification Phase via the FACI milestone. The document also maintains and further clarifies the software change procedure post the CDR milestone.

---

[115] Millard, "MIT Management Development Plan Meeting #6", 1-5
[116] J.V. Mutchler, "TRW Note No. 68-FMT-643. Apollo CMC/LGC Software Development Plan", 30 April 1968

In reality, no mission release was ever accepted according to these milestones. The *CDR* milestone was kept, but the *FACI* and the *CARR* milestones were not used. The final *Flight Software Readiness Review (FSSR)* milestone was kept as the only official meeting for accepting the release for a particular flight. The function on the *FACI* (to review test plans and instigate formal change control on requirements) and the *CARR* (to formally accept the release and to proceed to manufacturing) was accomplished by more dynamic regular Software Development Plan meetings between MIT and NASA.[117]

Once the GSOP had been approved, any subsequent changes had to be managed carefully, which is another important Software Engineering practice. This was first raised in a document in November 1967 where NASA/MSC committed to "develop and document a change procedure to assure the proper control and understanding of PCR's".[118] Previous to this, change control was ad hoc. Based on the guidelines in *Apollo CMC/LGC Software Development Plan*, NASA and MIT put in place a formal change control procedure around three types of artifacts: a Program Change Request (PCR), a Program Change Notice (PCN) and an Anomaly Report (AR).

A group within NASA, with representation from MIT, called the Apollo Spacecraft Software Configuration Control Board (ASSCCB), initiated and approved specific change requests, which were tracked along with the GSOP that they impacted. A PCR is a request for a change that impacts the GSOP. There was a myriad of underlying reasons for the change, such as input from NASA, MIT or a mission astronaut, either because the GSOP was incorrect or was nebulous. The PCR process started around November 1967.[119] The first GSOP revision that shows lists of approved PCRs is July 1968.[120] An example of an approved PCR is *PCR 210* from 1968 that requests a change to increase the DPS (Descent Propulsion System) Throttle Recovery Limit and to move that configuration from fixed memory to erasable memory.[121]

A PCN is a notification that a change has been made, either of low impact, such as documentation updates or to unblock program development in a timely manner. An example of an approved PCN

---

[117] Johnson and Giller, "R-700, Vol V, The Software Effort", 96
[118] Millard, "MIT Management Development Plan Meeting #6", 2
[119] Millard, "MIT Management Development Plan Meeting #6", 1 explicitly mentions PCR #4, so PCR #1 was sometime before this.
[120] Anonymous, "R-557 Guidance System Operations Plan For Manned LM Earth Orbital Missions Using Sundance (Rev. 306). Section 4 PGNCS Operational Modes (Rev 3)", MIT Instrumentation Laboratory, January 1969, v
[121] F.V. Bennett, "Apollo Spacecraft Software Configuration Control Board, Program Change Request 210", 11 June 1968, accessed 9 January 2025, https://www.ibiblio.org/apollo/Documents/PCR-210.pdf

is *PCN 497* that notifies a change in how the lunar landing braking phase (P63) calculates the throttle up time of the descent engine.[122]

The Software Control Board (SCB) was responsible for reviewing and approving PCRs and PCNs. For the example above, PCR 210 was approved at the *19th Software Control Board* meeting on 27 June 1968.[123] Once the software is released for manufacturing of the rope, further testing may uncover problems either with the rope manufacture or the functionality of the software in the rope. These problems are described in Anomaly Reports. As fixing a problem in the manufactured rope is expensive, an anomaly was usually mitigated via a program note for the mission. If this was not possible, then a PCR or PCN is raised and approved by the SCB. Lower-level changes, which were deemed not to impact the GSOP could be approved solely by MIT, via their Assembly Control Board. An example of an Anomaly Report is *LNY 79* that identified a lunar module oscillation issue with the P64/P65 lunar landing programs, resulting in a program note to the Apollo 11 crew.[124] The problem was fixed, via a PCR in a future release, for the Apollo 12 crew.[125]

The change control process, as designed, did impose constraints on changes after a certain point and this could cause frustration with the developers at MIT. For example, Jim Miller, who developed the AGC simulators, remembers:

> "… the purpose being to find any more bugs that hadn't been found. And sometimes, when you found-- sometimes you would find bugs that just you felt had to be fixed. And usually, they could be. But once the program was delivered, this group of people, who were the only people in the world who NASA thought could write the software, suddenly were so stupid that no change they proposed could be plausible at all. And so anything you suggested, the answer was no. And you had to beg and so forth, for the interest of the project, to be allowed to remove this desperately bad thing".[126]

---

[122] G.W. Cherry, "Apollo Spacecraft Software Configuration Control Board, Program Change Notice 497", 15 July 1968, accessed 9 January 2025, https://www.ibiblio.org/apollo/Documents/PCR-497.pdf
[123] G.W. Cherry "Apollo Project Memo No. 1933. LM Program Items Discussed and Acted on at the 19th SCB Meeting", 27 June 1968, accessed 9 January 2025, https://www.ibiblio.org/apollo/Documents/Memo-SCB19_text.pdf , 3
[124] K.W. Krause, "MIT/IL Software Anomaly Report, LNY 79, 20 June 1969, accessed 21 May 2025, https://www.ibiblio.org/apollo/Documents/LNY-79.pdf
[125] George W. Cherry and Fred H. Martin, "MIT/IL Software Development Plan for LUMINARY 1B LGC Program (Final Issue)", MIT Instrumentation Laboratory, 22 August 1969, II-4 shows that PCR 840 fixed LNY 79
[126] MIT Video Productions, "Apollo Guidance Computer Project", 1:18:50 – 1:19:28

The testing levels in the *Apollo CMC/LGC Software Development Plan* were also not used exactly as defined. There were six levels of testing, not four. Level 1 and Level 2 testing matched the document. Level 3 was program level testing on the 'All Digital' and 'Hybrid' simulators. Each test case was included in the GSOP and was part of the acceptance process. Level 4 was sequencing testing across multiple programs, again using the 'All Digital' and 'Hybrid' simulators. When Level 4 testing was completed, the software was ready to be released to manufacturing. Level 5 was a repeat of all Level 3 and 4 testing before the actual release to manufacturing. Level 6 testing, which occurred after the software was released to manufacturing, established performance specifications and was oriented towards a particular launch.[127] A reasonably good example of an L4 test plan can be found at *LUMINARY Memo #30*.[128]

Future versions of the Lunar Module software, SUNDANCE and then LUMINARY, included the programs for navigating down to the Moon's surface, launching from the Moon's surface and rendezvousing with the Command Module in lunar orbit. The strategic approach to landing on the Moon, on which this software is based, is described in detail in *Apollo lunar module landing strategy* from 1966.[129] This is a good example of a Software Engineering requirements document, as it details exactly how to get the lunar surface safely. It is the basis of the lunar module programs: P63 (Braking Phase), P64 (Final Approach Phase), P65 (Automatic Landing Phase) and P66 (Semi-Automatic Landing Phase).[130] SUNDANCE was used in Apollo 9. LUMINARY was used in Apollo 10, Apollo 11 and all future missions.

---

[127] Johnson and Giller, "R-700, Vol V, The Software Effort", 84-86
[128] J. Kernan and C. Schulenberg, "LUMINARY Memo #30, LUMINARY Level 4 Test Plan", MIT Instrumentation Laboratory, 7 June 1968
[129] Donald C. Cheatham, Floyd V. Bennett, and Theoretical Mechanics Branch. "Apollo lunar module landing strategy." in *Apollo Lunar Landing Symposium* 1966, 175-240.
[130] Phill Parker, "The Apollo On-board Computers", Apollo Flight Journal, accessed 16 May 2025, https://www.nasa.gov/history/afj/compessay.html . For an example of the underlying code, P63 Braking Phase is at "THE_LUNAR_LANDING.agc.html", The Virtual AGC Project GitHub repository, accessed 16 May 2025, https://www.ibiblio.org/apollo/listings/LUM69R2/THE_LUNAR_LANDING.agc.html

4 Software Engineering Matures (1968–1971)

In this chapter, we describe the culmination of the program, when the AGC was used to successfully land six manned missions on the Moon. We also describe the NATO conferences in 1968 and 1969 and argue that NASA and MIT had already resolved a lot of the issues raised at these conferences.

As development proceeded through 1968, the processes for defining requirements, tracking progress and change control across many releases were bedded in. While there were some teething problems at the start of 1968, by the end of the year, the process was running quite well. For example, in March 1968, Howard Tindall wrote “I guess I am attacking the old ‘MIT me’ in stating that we are seriously handicapped by having no reliable definition of the Luminary lunar surface and ascent programs”.[131] In May 1968, he raised issues with the PCR process, “That strikes me as a real improvement in the program but it mystifies me as how it go[t] changed without a PCR or PCN, or even letting anyone know”.[132] However, towards the end of the year, most of Tindall’s memos are about “mission technique” meetings, which define specific requirements for the C missions (demonstrate Command and Service Module, crew performance and rendezvous capability), D mission (demonstrate Lunar Module and crew performance in Earth Orbit), F mission (demonstrate Lunar Module and crew performance in Lunar Orbit), and G mission (manned lunar landing demonstration). Although these requirements are fed to MIT and result in changes to the software, Tindall’s memos later in the year barely mention MIT, implying that the process is running smoothly. A good example of this is the ASSCCB meeting from 8 October 1968, which describes various PCRs that were reviewed and approved at this meeting, without any significant contention.[133]

---

[131] Howard W. Tindall, “68-PA-T-48A. Ascent Phase Mission Techniques meeting - February 27”, 4 March 1968, accessed on 7 May 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

[132] Howard W. Tindall, “68-PA-T-106A. Spacecraft computer program newsletter”, 24 May 1968, accessed on 7 May 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

[133] Howard W. Tindall, “68-FM-T-225. Results of' the October 8 Apollo Spacecraft Software Configuration Control Board (ASSCCB) meeting”, 16 October 1968, accessed on 7 May 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf This document approves PCR 547, which is J. Shillingford, “Apollo Spacecraft Software Configuration Control Board, Program Change Request 547”, 3 September 1968, accessed 7 May 2025, https://www.ibiblio.org/apollo/Documents/PCR-547.pdf

Also, in October 1968, the Final Report from the Apollo Guidance Software Task Force was published and was quite positive in its conclusions.[134] In summary, it concluded that the situation at the time of publishing was good, no major improvements needed to be made, complex software required an ongoing high degree of communication and that schedule control and visibility was adequate. This was a positive reflection on the pain and effort that MIT and NASA had gone through during the development of the AGC.

Between 1968 and the early part of 1971, MIT delivered high quality, reliable AGC software for eleven manned missions, including nine versions of the Command Module software and seven versions of the Lunar Module software.[135] In agreement with NASA, the naming convention was changed to use 'C' words for Command Module software and 'L' words for Lunar Module software.[136] Hence the move to names like COLOSSUS and COMANCHE for the Command Module and LUMINARY for the Lunar Module.

Margaret Hamilton continued to assume an active leadership position in the CSM software development group during this time. From an organisational chart from February 1969, she was the Director of the CSM Programming group within the Mission Program Development Division (23B).[137] Hamilton's peer, J. Kernan led the LM group. Dan Lickly was the overall owner of 23B at this time.

The first successful Moon landing was Apollo 11 on 20 July 1969 when Niall Armstrong and Buzz Aldrin became the first men to walk on the Moon. The AGC Command Module software was COLOSSUS IIA, released in April 1969 and the Lunar Module software release was LUMINARY IA, released in June 1969. The software performed admirably, but there was significant uncertainty during the landing sequence when "1201" and "1202" program alarms happened. These indicated that the computer was overloaded and that it was concentrating only on the highest priority tasks, meaning the landing itself. The computer had behaved correctly, and the problem was later

---

[134] W.G. Heffron, "NASA CR-97638, Distribution of the Final Report of the Apollo Guidance Software Task Force", Bellcomm, Inc. 7 October 1968, 6-7
[135] Hall, "R-700 Vol III, Computer Subsystem", 153
[136] Johnson and Giller, "R-700, Vol V, The Software Effort", 6
[137] Anonymous, "Apollo Organization", MIT Instrumentation Laboratory, 1 February 1969, accessed 3 July 2025, https://www.doneyles.com/LM/ORG/index.html , 5

diagnosed as a misplaced switch in the Lunar Module related to the rendezvous radar.[138] This issue was fixed for the Apollo 12 software via the existing PCR process.[139]

After Apollo 11, Dan Lickly left the Instrumentation Laboratory and Margaret Hamilton assumed overall leadership of 23B sometime between August and September 1969.[140] She was to lead this group until the end of the Apollo program.

Feature development and bug fixes continued through Apollo 15. The final software for the Command Module and Lunar Module was released in February and March 1971, respectively. As the software had become very stable by this point, features and bug fixes reduced to the point where the exact same software flew on the final three missions: Apollo 15, 16 and 17. In a memo approving this approach from August 1971, Howard Tindall wrote in the margin "Behold these people, for they have created perfection !!!".[141]

As the 1960s progressed and the available compute power increased, the problems with building large scale software projects that MIT and NASA had already uncovered became more evident to other parts of the industry. As Dijkstra reflected on the software crisis of the 1960s "and now we have gigantic computers, programming has become an equally gigantic problem".[142] To address this growing concern, NATO organized two conferences, one in 1968 in Garmisch, Germany and the other in 1969 in Rome, Italy.[143] The conferences are generally credited with helping establish

---

[138] Apollo 11 Mission Evaluation Team, "16.2.5 Computer Alarms During Descent" in *Apollo 11 Mission Report MSC-00171*, NASA, Manned Spacecraft Center, Houston, Texas, November 1969, 16-12 to 16-14

[139] D. Eyles, "Apollo Spacecraft Software Configuration Control Board, Program Change Request 848, Prevent RR ECDUs from Stealing LGC Memory Cycles", 23 July 1969, accessed 2 July 2025, https://www.ibiblio.org/apollo/Documents/PCR-848.pdf Note this is only three days after the problem occurred during the Apollo 11 landing. The PCR was implemented before 6 August 1969 as per C. Schulenberg, "LUMINARY Memo #103", MIT Instrumentation Laboratory, 6 August 1969, 2. The code that implements the fix for Apollo 12 can be found at "T4RUPT_PROGRAM.agc.html", The Virtual AGC Project GitHub repository, accessed 2 July 2025, https://www.ibiblio.org/apollo/listings/Luminary116/T4RUPT_PROGRAM.agc.html#525241555443484B

[140] Margaret Hamilton, "R-567 Guidance System Operations Plan for Manned LM Earth Orbital and Lunar Missions using Program LUMINARY 1B (Rev. 116) Section 4 PGNCS Operational Modes (Rev. 6)", MIT Instrumentation Laboratory, September 1969, is signed off by Margaret Hamilton, who has the title "Director, Mission Program Development Apollo Guidance and Navigation Program". Margaret Hamilton, "COMANCHE 55 AGC Program Listing", 28 March 1969, 1, accessed 12 October 2025, https://archive.org/details/Comanche55J2k60/mode/1up . This is the Apollo 11 CSM software and lists Dan Lickly as the overall owner of 23B on 28 March 1969, with Margaret Hamilton responsible for the CSM group. Lickly left MIT to start Intermetrics in 1969. However, it is reasonable to assume that he stayed with MIT until Apollo 11 returned on 24 July 1969. Therefore, the time window for Hamilton to take over 23B is between August and September 1969.

[141] Howard W. Tindall, "FA-115, Flight ropes for Apollo 16 and 17", 17 August 1971, 1, accessed on 2 July 2025, https://www.ibiblio.org/apollo/Documents/16_17_flight_ropes.pdf

[142] Edsger W. Dijkstra, "The Humble Programmer." *Communications of the ACM* 15, no. 10 (1972): 861.

[143] Peter Naur and Brian Randell, eds. *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968,* Brussels, NATO Scientific Affairs Division, 1969; J. N. Buxton and

software engineering as a recognized discipline. They discussed similar problems that had already occurred at MIT, such as the lack of clear requirements and the inability to deliver on schedule. They discussed solutions to these problems, including introducing education on engineering practices, and introducing practices such as clearly defined requirements, documented designs, testing plans, acceptance plans and change management. MIT were not present at either conference, but the Apollo program did participate in the 1969 conference where a paper discussing the ground control software in support of mission control was presented and discussed.[144] It is unclear why MIT did not participate.

Most of the problems related to the "software crisis" had been experienced by MIT during the development of the Apollo Guidance Computer.[145] Similarly, through trial and error, and the imposition of a hard deadline to land a man on the Moon by the end of the 1960s, a lot of the postulated solutions had been implemented by NASA and MIT by the time of these conferences. However, the rest of the nascent industry did not know about them.

B. Randell, eds., *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27th to 31st October 1969*, Brussels, NATO Scientific Affairs Division, 1970

[144] Buxton and Randell, *Software Engineering Techniques*, 32-36

[145] Naur and Randell, *Software Engineering*, 70-71

Conclusion

The goal of this dissertation is to examine MIT's software development efforts for the Apollo Guidance Computer and chart how Software Engineering practices were created by these teams during the period of 1961- 1971.

The first chapter provides context on the politics and technology of the early 1960s, a background to Margaret Hamilton and an overview of the AGC hardware development. It then describes the complexity of the overall software effort and examines the early operating system development and the first release of the software, which was done without any formal requirements document, design document or user acceptance document. It compares this approach to the computer's hardware development, which was much more engineering focused effort. On the positive side, it details the approach to a testing strategy, which was encouraging. However, these years did not have any recognizable Software Engineering practices.

The *Initial Inroads into Software Engineering (1964–1966)* chapter details improvements in documentation for the next set of releases, which included the first unmanned mission. This is the first time we see the Guidance and Navigation Systems Operations Plan (GSOP), which in later years was to become the preeminent document for every mission. We also see improvements in the testing strategy and NASA's early attempts at managing large scale software projects. These all point in the right direction for Software Engineering. However, the size and complexity of the project had pushed MIT's informal approach to its limit.

Crisis and the Emergence of Software Engineering (1966-1968) details the difficulties in 1966 where schedule pressure and delivery delays caused significant friction between NASA and MIT. NASA again attempted to define how to manage large scale software projects, but it was not until mid 1967 that this was agreed with MIT, who also put a new organisational structure in place. It is during this period that the GSOP document becomes more formal and effectively becomes the de-facto functional requirements and user acceptance document. Outside of the GSOP, a formal change control process was put in place, which controlled how changes were made after a certain point. It was also during this period that Margaret Hamilton first started using the term "Software Engineering".

Lastly, *Software Engineering Matures (1968–1971)* details how the agreed processes started to pay dividends and the level of tension between NASA and MIT decreased. MIT became an efficient Software Engineering machine and successfully delivered high quality software for the six manned missions up to and including Apollo 17. It also outlines the two NATO conferences which are also credited with creating the discipline of Software Engineering. The chapter argues that MIT had already experienced the "software crisis" discussed at these conferences during the AGC software development effort and had already defined Software Engineering solutions to solve these crises. However, these solutions had not been widely disseminated.

In comparison to the practices of Software Engineering defined in the Introduction, the approach taken by MIT and NASA is impressive. On the positive side, the Guidance and Navigation Systems Operations Plan (GSOP) document forms the basis of the following steps: *Requirements Analysis and Definition*, *Testing Strategy* and *User Acceptance*. While, during the early phases, the GSOP was written after the release, this morphed in later phases to be written well before the software was finished, which is more correct. *Release Management* is covered well in the numerous software releases and the manufacturing of core memory ropes. The *Software Maintenance* step is covered well in the processes around Program Change Requests, Program Change Notices and Anomaly Reports.

On the negative side, there is no documentation on *Design and Architecture*, although there is some after the fact documentation. There is little documentation on *Implementation processes*, such as coding standards and coding style. While the language choice was constrained to the AGC Basic and Interpreted assembly language, there is no documentation on how to write consistent, maintainable code. There are informal guidelines on commenting on the code, but nothing else.

The most advanced aspect of the AGC Software Engineering was the *Testing Strategy*, both for the creation of the 'All Digital' and Hybrid simulators and the creation of six levels of testing as part of the acceptance process. While the initial approach was created due to the lack of hardware availability, this grew into a highly flexible, very advanced, testing strategy with full mock-ups of the Command Module and the Lunar Module and the ability to test every conceivable scenario.

The "software crisis" described at the NATO conferences in 1968 and 1969 had already happened at MIT in 1966 and the solutions for these crises had been put in place. The hard deadline of the

end of the decade imposed by Kennedy's goal forced MIT and NASA to define Software Engineering practices, well before any other institution. This resulted in a set of software that helped meet Kennedy's goal of landing a man on the Moon and returning him to earth. As Hall points out "The fact remains that Apollo computers successfully guided nine lunar missions and five Earth-orbital missions without a failure".[146] However, there is limited evidence of MIT communicating their Software Engineering approach after Apollo completed. This is an area for future study. Most of the key individuals remained with MIT to work on the software for the Space Shuttle program. After leaving MIT in 1976, Margaret Hamilton started a company called Higher Order Software. She also founded Hamilton Technologies in 1986. Both companies were involved in the detection and avoidance of errors in early software development.

It was not until much later in the 20th century that broad credit was given to MIT for the technological achievement they accomplished in building the Apollo Guidance Computer based on early 1960s technology. Margaret Hamilton was one of several key people in this project. She joined the effort in April 1965, took over leadership of the CSM software group in 1967 and led the overall onboard software effort from Sep 1969 until the end of Apollo. She has become an iconic figure within Software Engineering. In 2003, she was awarded NASA's Exceptional Space Act Award and in 2016, she received the Presidential Medal of Freedom from President Obama, who remarked:

> "Three minutes before Armstrong and Aldrin touched down on the Moon, *Apollo 11*'s lunar lander alarms triggered: red and yellow lights across the board. Our astronauts didn't have much time. But thankfully, they had Margaret Hamilton. A young MIT scientist—and a working mom in the sixties—Margaret led the team that created the onboard flight software that allowed the Eagle to land safely. And keep in mind that, at this time, software engineering wasn't even a field yet. There were no textbooks to follow, so, as Margaret says, "There was no choice but to be pioneers." Luckily for us, Margaret never stopped pioneering".[147]

---

[146] Hall, *Journey to the moon*, 120

[147] Barack Obama, "Remarks on Presenting the Presidential Medal of Freedom", 22 November 2016, accessed 3 July 2025, https://www.presidency.ucsb.edu/documents/remarks-presenting-the-presidential-medal-freedom-15

We owe a lot to MIT and to NASA as they were pioneers in the emergence of Software Engineering. What they created during the 1960s would be recognizable to every Software Engineer today.

Areas for future work include researching the AGC testing strategy, the simulators that MIT built, the companies that were later founded by Margaret Hamilton and how all of this influenced future Software Engineering concepts such as Agile Development and Test-Driven-Development. Another area is applying modern automated testing techniques to old code bases, such as the Apollo Guidance Computer software, to quantify the correctness of that software and potentially identify unknown bugs in that software. And finally, the development and institutionalisation of Software Engineering principles post the Apollo Guidance Computer, and the NATO conferences is an area for further research.

Appendix: Apollo Guidance Computer Software Releases

A detailed breakdown of the AGC software releases, including release name and version identifier is in the table below.[148]

Release Date / Mission Date	Mission Identifier	Mission Type[149]	Mission Name	Manned	CSM AGC Version	LM AGC Version
February 1964 / NA					ECLIPSE	
July 1964 / NA					SUNRISE/33	
January 1966 25 August 1966	AS-202			Unmanned	CORONA/261	
July 1966 Cancelled (Apollo 1 Fire)	AS-204A		Apollo 1	Manned	SUNSPOT	
November 1966 9 November 1967	AS-501	A-1	Apollo 4	Unmanned	SOLARIUM/55	
October 1967 22 January 1968	AS-204		Apollo 5	Unmanned		SUNBURST/120
November 1966 4 March 1968	AS-502	A-2	Apollo 6	Unmanned	SOLARIUM/55	
3 March 1966 27 May 1968	LTA-8			Manned		AURORA/85
13 Mar 1967 16 June 1968	2TV-1			Manned	SUNDIAL/E	
February 1968 11 October 1968		C	Apollo 7	Manned	SUNDISK/282	
August 1968 21 December 1968		C	Apollo 8	Manned	COLOSSUS I/237	

[148] "Command-Module Flight Software", The Virtual AGC Project, accessed 4 July 2025, https://www.ibiblio.org/apollo/Colossus.html#gsc.tab=0 ; "Lunar Module Flight Software", The Virtual AGC Project, accessed 4 July 2025, https://www.ibiblio.org/apollo/Luminary.html#gsc.tab=0

[149] "Apollo Program History", March To The Moon, accessed 14 May 2025, https://tothemoon.im-ldi.com/about/apollo_history

Release Date / Mission Date	Mission Identifier	Mission Type[149]	Mission Name	Manned	CSM AGC Version	LM AGC Version
CM Oct/1968 LM October 1968 3 March 1969		D	Apollo 9	Manned	COLLOSSUS IA/249	SUNDANCE/306
CM April 1969 LM April 1969 18 May 1969		F	Apollo 10	Manned	COLOSSUS II/MANCHE45rev2	LUMINARY I/69 Rev 2
CM April 1969 LM June 1969 16 July 1969		G	Apollo 11	Manned	COLOSSUS IIA/COMANCHE55	LUMINARY IA/LMY99 rev 1
CM July 1969 LM August 1969 14 November 1969		H-1	Apollo 12	Manned	COLOSSUS IIB/COMANCHE67	LUMINARY IB/ 116
CM December 1969 LM February 1970 11 April 1970		H-2	Apollo 13	Manned	COLOSSUS IIC/MANCHE72 rev 3	LUMINARY IC/LM131 rev 1
CM May 1970 LM September 1970 31 January 1971		H-3	Apollo 14	Manned	COLOSSUS IID/COMANCHE108	LUMINARY ID/178
CM February 1971 LM March 1971 26 July 1971		J-1	Apollo 15	Manned	COLOSSUS III/ARTEMIS72	LUMINARY IE/210
CM February 1971 LM March 1971 16 April 1972		J-2	Apollo 16	Manned	COLOSSUS III/ARTEMIS72	LUMINARY IE/210
CM February 1971 LM March 1971 7 December 1972		J-3	Apollo 17	Manned	COLOSSUS III/ARTEMIS72	LUMINARY IE/210

Bibliography

Primary Sources

610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology. S.l.: IEEE, 1990.

Alonso R. and J.H. Lanning, Jr., H. Blair Smith. “E-1077, Preliminary Mod 3C Programmers Manual”, MIT Instrumentation Laboratory, November 1961.

Alonso R. and J.H. Lanning, Jr. “R-276, Design Principles for a General Control Computer”, MIT Instrumentation Laboratory, April 1960.

Anonymous. “Apollo Guidance Computer Information Series, Issue 15, Block I Apollo Guidance Computer Subsystem, FR-2-115”, 27 March 1964

Anonymous, “Apollo Organization”, MIT Instrumentation Laboratory, May 1962, accessed 3 July 2025, https://www.ibiblio.org/apollo/Documents/ApolloOrg-1962-05.jpg

Anonymous, “Apollo Organization”, MIT Instrumentation Laboratory, May 1967, accessed 3 July 2025, https://wehackthemoon.com/sites/default/files/styles/hero_extra_large_1x_2000x_/public/media/photograph/45838.jpg?itok=ECT1UPEe

Anonymous, “Apollo Organization”, MIT Instrumentation Laboratory, 30 September 1968, accessed 12 October 2025, https://www.ibiblio.org/apollo/Documents/ApolloOrg-Group-23B-only-1968-09.jpg

Anonymous, “Apollo Organization”, MIT Instrumentation Laboratory, 1 February 1969, accessed 3 July 2025, https://www.doneyles.com/LM/ORG/index.html

Greene, K.W., "MIT Management Development Plan Meeting #4", MIT Instrumentation Laboratory, 7 Nov 1967, accessed 12 October 2025, https://www.ibiblio.org/apollo/Documents/mit_dev_plan_meeting_04.pdf

Anonymous. “MSC-A-R-66-5, Postlaunch Report for Mission AS-202 (Apollo Spacecraft 011)”, NASA Manned Spacecraft Center, 12 October 1966

Anonymous. “R-527 (Rev 2) Guidance System Operations Plan for Unmanned LM Earth Orbital Missions using Program SUNBURST, Section 7 G&N Error Analyses”, MIT Instrumentation Laboratory, December 1967

Anonymous. “R-537 Guidance and Navigation System Operations Plan Apollo Mission 501”, MIT Instrumentation Laboratory, July 1966

Anonymous. “R-557 Guidance System Operations Plan For Manned LM Earth Orbital Missions Using Sundance (Rev. 306). Section 4 PGNCS Operational Modes (Rev 3)”, MIT Instrumentation Laboratory, January 1969

Anonymous, “R-700, MIT’s Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume II, Optical, Radar and Candidate Subsystems”, MIT Charles Stark Draper Laboratory, March 1972

Anonymous. “Report E-1236. Monthly Technical Progress Report. Project Apollo Guidance and Navigation Program. Period Jun 11, 1962 through July 17, 1962”, MIT Instrumentation Laboratory, 17 July 1962

Apollo 4 Mission Evaluation Team. “Apollo 4 Mission Report”, NASA Manned Spacecraft Center, Houston, Texas, 7 January 1968

Apollo 5 Mission Evaluation Report, “Apollo 5 Mission Report, MSC-PA-R-68-7”, NASA Manned Spacecraft Center, Houston, Texas, 27 March 1968

Apollo 11 Mission Evaluation Team, “16.2.5 Computer Alarms During Descent” in *Apollo 11 Mission Report MSC-00171*, NASA, Manned Spacecraft Center, Houston, Texas, November 1969

Battin, R. and R.Crisp, A.Green et al. “R-467, The Compleat Sunrise, Being a Description of Program SUNRISE, (SUNRISE 33 – NASA DWG #1021102)”, MIT, September 1964

Bennett, F.V. “Apollo Spacecraft Software Configuration Control Board, Program Change Request 210”, 11 June 1968, accessed 9 January 2025, https://www.ibiblio.org/apollo/Documents/PCR-210.pdf

Boston Globe, 18 October 1964, 109, accessed 30 June 2025 https://bostonglobe.newspapers.com/image/433781571/

Brock, David C. “Oral History of Margaret Hamilton”, Computer History Museum, CHM Reference number: X8186.2017, accessed 16 January 2025, https://archive.computerhistory.org/resources/access/text/2022/03/102738243-05-01-acc.pdf

Burkey, Ronald S. “Apollo Program MIT technical documents from JSC”, accessed 6 May 2025, https://www.ibiblio.org/apollo/NARA-SW/IndexOfNaraBox209G.html

Buxton, J. N. and B. Randell, eds. *Software Engineering Techniques: Report on a Conference Sponsored by the NATO Science Committee, Rome, Italy, 27th to 31st October 1969*, Brussels, NATO Scientific Affairs Division, 1970

Cheatham, Donald C., Floyd V. Bennett, and Theoretical Mechanics Branch. "Apollo lunar module landing strategy." in *Apollo Lunar Landing Symposium* 1966, 175-240.

Cherry, G.W. “Apollo Project Memo No. 1933. LM Program Items Discussed and Acted on at the 19th SCB Meeting”, 27 June 1968, accessed 9 January 2025, https://www.ibiblio.org/apollo/Documents/Memo-SCB19_text.pdf

Cherry, G.W. “Apollo Spacecraft Software Configuration Control Board, Program Change Notice 497”, 15 July 1968, accessed 9 January 2025, https://www.ibiblio.org/apollo/Documents/PCR-497.pdf

Cherry, George W. and Fred H. Martin. “MIT/IL Software Development Plan for LUMINARY 1B LGC Program (Final Issue)”, MIT Instrumentation Laboratory, 22 August 1969

Dahlen, John M. and Albrecht Kosmala, Daniel J. Lickly, John T. Shillingford, Balraj Sokkappa. “R-477 Guidance and Navigation System Operations Plan, Apollo Mission 202”, January 1965

Dijkstra, Edsger W. "The Humble Programmer." *Communications of the ACM* 15, no. 10 (1972): 859-866.

Eyles, D. “Apollo Spacecraft Software Configuration Control Board, Program Change Request 848, Prevent RR ECDUs from Stealing LGC Memory Cycles”, 23 July 1969, accessed 2 July 2025, https://www.ibiblio.org/apollo/Documents/PCR-848.pdf

Felleman, Philip G. “E-2066 Hybrid Simulation of the Apollo Guidance Navigation and Control System”, MIT Instrumentation Laboratory, December 1966.

Fisk, Dale. *Programming with Punched Cards*, 2005, accessed 28 February 2025 https://web.archive.org/web/20090325223903/https://www.columbia.edu/acis/history/fisk.pdf

Glick, F.K. and Femino, S.R. “E-2475 A Comprehensive Digital Simulation for the Verification of Apollo Flight Software”, MIT Charles Stark Draper Laboratory, January 1970.

Hall, Eldon C. “R-410 General Design Characteristics of the Apollo Computer”, MIT Instrumentation Laboratory, May 1963

Hall, Eldon C. “R-700, MIT’s Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume III, Computer Subsystem”, MIT Charles Stark Draper Laboratory, August 1972

Hamilton, Margaret, “COMANCHE 55 AGC Program Listing”, MIT Instrumentation Laboratory, 28 March 1969, accessed 12 October 2025, https://archive.org/details/Comanche55J2k60/mode/1up

Hamilton, Margaret, “R-567 Guidance System Operations Plan for Manned LM Earth Orbital and Lunar Missions using Program LUMINARY 1B (Rev. 116) Section 4 PGNCS Operational Modes (Rev. 6)”, MIT Instrumentation Laboratory, September 1969

Hamilton, Margaret. “The Language as a Software Engineer”, ICSE 2018 Plenary Sessions, 31 May 2018. Video, 25:26, https://www.youtube.com/watch?v=ZbVOF0Uk5lU

Hand, James A. “R-700, MIT’s Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 9-4065, Volume I, Project Management, Systems Development, Abstracts and Bibliography”, MIT Charles Stark Draper Laboratory, October 1971

Heffron W.G. “NASA CR-97638, Distribution of the Final Report of the Apollo Guidance Software Task Force”, Bellcomm, Inc. 7 October 1968.

Hopkins, Albert and Ramon Alonso, Hugh Blair Smith. “R-393, Logical Description for the Apollo Guidance Computer (AGC 4)”, 5 March 1963

Johnson, Madeline S. and Donald R. Giller. “R-700, MIT’s Role in Project Apollo, Final Report on Contracts NAS 9-153 and NAS 0-4065, Volume V, The Software Effort”, MIT Charles Stark Draper Laboratory, March 1971

Keese W.M. and B. H. Liebowitz, W.J. Martin et al., “Management Procedures In Computer Programming Apollo – Interim Report”, Bellcomm Inc, 30 November 1964

Kernan, J. and C. Schulenberg. “LUMINARY Memo #30, LUMINARY Level 4 Test Plan”, MIT Instrumentation Laboratory, 7 June 1968

Krause, K.W. “MIT/IL Software Anomaly Report, LNY 79, 20 June 1969, accessed 21 May 2025, https://www.ibiblio.org/apollo/Documents/LNY-79.pdf

Liebowitz, B.H. and C.S. Sheppard, E.B Parker III. “Procedures for Management Control of Computer Programming in Apollo”, Bellcomm Inc., 28 September 1966

Lorenz, E.N., “The statistical prediction of solutions of dynamical equations”. *Proc. Int. Symp. on Numerical Weather Prediction*, Tokyo, Japan, Meteorological Society of Japan, 1962, 629–634.

Millard, R.C. “MIT Management Development Plan Meeting #6”, 21 November 1967

Miller, John E. and Ain Laats. “E-2397, Apollo Guidance and Control System Flight Experience”, MIT Instrumentation Laboratories, June 1969.

Mimno, Peter. “R-599 Digital Simulation Manual”, MIT Instrumentation Laboratory, January 1968.

MIT Video Productions. “Apollo Guidance Computer Project – MIT History Conference pt.2 – 2001”, 14 September 2001, https://www.youtube.com/watch?v=quIfco4RLCg

Mutchler, J.V. “TRW Note No. 68-FMT-643. Apollo CMC/LGC Software Development Plan”, 30 April 1968

Naur, Peter and Brian Randell, eds. *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968,* Brussels, NATO Scientific Affairs Division, 1969

Nevins, J. L. “MIT Instrumentation Laboratory, DG Memo No.88, Description and Status of AGC Program”, 20 January 1964

Obama, Barack “Remarks on Presenting the Presidential Medal of Freedom”, 22 November 2016, accessed 3 July 2025, https://www.presidency.ucsb.edu/documents/remarks-presenting-the-presidential-medal-freedom-15

Sampson, J.A. “Flight 501 Memo #8. Programming Changes from AS-202 to AS-501”, MIT Instrumentation Laboratory, 7 October 1968

Sampson, Jay. “Flight 501 Memo #10, Summary of Results of AS-501 Digital Simulations”, MIT Instrumentation Laboratory, 26 October 1966

Schulenberg, C. “LUMINARY Memo #103”, MIT Instrumentation Laboratory, 6 August 1969

Shillingford, J. “Apollo Spacecraft Software Configuration Control Board, Program Change Request 547”, 3 September 1968, accessed 7 May 2025, https://www.ibiblio.org/apollo/Documents/PCR-547.pdf

Sullivan, Madeline M. “Hybrid Simulation of the Apollo Guidance and Navigation System”, *Simulation*, Vol 7 No 1 July 1966.

The Guidance Software Validation Committee. "Apollo Guidance Software Development and Verification Plan”, 4 October 1967

The Virtual AGC Project, “Apollo and Gemini Document Library”, accessed 23 July 2023, https://www.ibiblio.org/apollo/links2.html#gsc.tab=0

The Virtual AGC Project, “Command-Module Flight Software”, accessed 4 July 2025, https://www.ibiblio.org/apollo/Colossus.html#gsc.tab=0

The Virtual AGC Project, “Lunar Module Flight Software”, accessed 4 July 2025, https://www.ibiblio.org/apollo/Luminary.html#gsc.tab=0

The Virtual AGC Project GitHub Repository. “SINGLE_PRECISION_SUBROUTINES.agc.html”, accessed 20 January 2025, https://www.ibiblio.org/apollo/listings/Comanche055/SINGLE_PRECISION_SUBROUTINES.agc.html

The Virtual AGC Project GitHub repository. “SUNBURST/120 FRESH_START_AND_RESTART.agc.html”, accessed 1 July 2025 https://www.ibiblio.org/apollo/listings/Sunburst120/FRESH_START_AND_RESTART.agc.html#464F524745544954

The Virtual AGC Project GitHub repository. “T4RUPT_PROGRAM.agc.html”, accessed 2 July 2025, https://www.ibiblio.org/apollo/listings/Luminary116/T4RUPT_PROGRAM.agc.html#525241555443484B

The Virtual AGC Project GitHub repository. “THE_LUNAR_LANDING.agc.html”, accessed 16 May 2025, https://www.ibiblio.org/apollo/listings/LUM69R2/THE_LUNAR_LANDING.agc.html

Tindall, Howard W. “66-FM1-59. Spacecraft computer program requirements for AS-207/208, AS-503, and AS-504”, 12 May 1966, accessed on 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “66-FM1-70. Spacecraft computer program status report”, 2 June 1966, accessed 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “66-FM1-100. Notes regarding the AS-207/208 Guidance Systems Operation Plan (GSOP) meeting with MIT”, 30 August 1966, accessed 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “66-FM1-124. Program Development Plans are coming!!”, 12 October 1966, accessed on 21 March 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “66-FM1-170. More interesting things about our work with MIT”, 28 November 1966, accessed 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “66-FM1-191. MIT’s digital computers are saturated until the IBM 360 becomes operational”, 22 December 1966, accessed 13 February 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “67-FM1-17. AS-206 Spacecraft Computer Program Newsletter”, 31 January 1967, accessed on 17 March 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “67-FM1-18. Spacecraft Computer Program Development Newsletter”, 27 February 1967, accessed on 17 March 2025, https://www.ibiblio.org/apollo/Documents/1967_tindallgrams.pdf .

Tindall, Howard W. “67-FM1-23. Summary of what needs to be done to develop flight confidence in the spacecraft computer programs”, 23 March 1967, accessed on 17 March 2025, https://www.ibiblio.org/apollo/Documents/1967_tindallgrams.pdf .

Tindall, Howard W. “67-FM1-39. A new spacecraft computer program development working philosophy is taking shape”, 17 May 1967, accessed on 13 January 2025, https://web.mit.edu/digitalapollo/Documents/Chapter7/tindallgrams.pdf

Tindall, Howard W. “68-PA-T-48A. Ascent Phase Mission Techniques meeting - February 27”, 4 March 1968, accessed on 7 May 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “68-PA-T-106A. Spacecraft computer program newsletter”, 24 May 1968, accessed on 7 May 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “68-FM-T-225. Results of' the October 8 Apollo Spacecraft Software Configuration Control Board (ASSCCB) meeting”, 16 October 1968, accessed on 7 May 2025, https://www.ibiblio.org/apollo/Documents/tindallgrams02.pdf

Tindall, Howard W. “FA-115, Flight ropes for Apollo 16 and 17”, 17 August 1971, 1, accessed on 2 July 2025, https://www.ibiblio.org/apollo/Documents/16_17_flight_ropes.pdf

Trageser, Milton B. “A recoverable interplanetary space probe”, *Astronautics*, Volume 5, Issue 5, May 1960

Turing, A. M. "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society* s2-42, no. 1 (1937): 230-265.

Secondary Sources

Ambika, G. "Ed Lorenz: Father of the 'Butterfly Effect'." *Resonance* 20, no. 3 (2015): 198-205.

Bilstein, Roger E. *Stages to Saturn: A Technological History of the Apollo/Saturn Launch Vehicles*. Washington, DC: NASA SP-4206, 1980.

Booch, Grady. "The History of Software Engineering." *IEEE Software* 35, no. 5 (2018): 108-114.

Brooks, Courtney G. and James M. Grimwood, Loyd S. Swenson. "Preparations for the First Manned Apollo Mission" in *Chariots for Apollo: A History of Manned Lunar Spacecraft*, NASA Special Publication-4205, NASA History Series, 1979. Accessed 10 February 2025, https://solarviews.com/history/SP-4205/ch8-7.html

Campbell-Kelly, Martin. *Computer: A History of the Information Machine*. 3rd ed. Boulder: Westview Press, 2014.

Garner, Robert and Rick Dill, "The Legendary IBM 1401 Data Processing System." *IEEE Solid State Circuits Magazine* 2, no. 1 (2010): 28-39.

Hack The Moon. "The First Apollo Mission Contract goes to…". Accessed 8 May 2025, https://wehackthemoon.com/people/first-apollo-mission-contract-goes

Hall, Eldon C. *Journey to the moon: the history of the Apollo guidance computer*. Aiaa, 1996.

Hancock, James Rubio. "Margaret Hamilton, the programming pioneer who took Apollo to the Moon", El Pais, 25 December 2014, accessed 16 January 2025, https://verne.elpais.com/verne/2014/12/11/articulo/1418314336_993353.html

Hattis, Philip D. "How Doc Draper Became The Father Of Inertial Guidance", *Advances in the Astronautical Sciences AAS/AIAA Guidance, Navigation and Control* 2018, volume 164.

Launius, Roger D. "Interpreting the Moon Landings: Project Apollo and the Historians." *History and Technology* 22, no. 3 (2006): 225-255.

Light, Jennifer S. "When Computers were Women." *Technology and Culture* 40, no. 3 (1999): 455-483

Logsdon, John M. *The Decision to Go to the Moon: Project Apollo and the National Interest.* Cambridge, MA: The MIT Press, 1970.

McDougall, Walter A. ... *The Heavens and the Earth: A Political History of the Space Age*. New York: Basic Books, 1985.

March To The Moon. “Apollo Program History”, accessed 14 May 2025, https://tothemoon.im-ldi.com/about/apollo_history

Maurer, Richard. *The Woman in the Moon: How Margaret Hamilton Helped Fly the First Astronauts to the Moon*. Roaring Brook Press, 2023.

Mieczkowski, Yanek. *Eisenhower's Sputnik Moment: The Race for Space and World Prestige*. 1st ed. Ithaca: Cornell University Press, 2013.

Mindell, David A. *Digital Apollo: Human and Machine in Spaceflight*. 1st ed. Cambridge, MA: MIT Press, 2008;2011

Murray, Charles and Catherine Bly Cox. *Apollo: The Race to the Moon*. New York: Simon and Schuster, 1989.

O’Brien, Frank, ed. *The Apollo guidance computer: Architecture and operation*. New York, NY: Praxis, 2010.

O’Regan, Gerard. *Brief History of Computing*. 3;3rd 2021;Third; ed. Cham: Springer, 2021

Parker, Phil. “The Apollo On-board Computers”, Apollo Flight Journal, accessed 16 May 2025, https://www.nasa.gov/history/afj/compessay.html .

Shetterly, Margot Lee. *Hidden Figures: The American Dream and the Untold Story of the Black Women Mathematicians Who Helped Win the Space Race.* New York: HarperCollins, 2016.

Shorey, Samantha and Daniela K. Rosner. "A voice of process: re-presencing the gendered labor of Apollo innovation." *communication+ 1* 7, no. 2 (2019).

Tomayko, James E. "Computers in spaceflight: the NASA experience." *Kent, Allen; Williams, James G., eds. Encyclopedia of Computer Science and Technology* 18, no. NAS 1.26: 182505 (1988).

Tylko, John. “MIT and navigating the path to the moon”, *AeroAstro*, 2009, accessed 19 January 2025, https://web.mit.edu/aeroastro/news/magazine/aeroastro6/mit-apollo.html