# Defining a System

Grant Martin glm2367

October 20, 2021

Throughout the various sciences and a multitude of perspectives, the term system is used quite liberally in various ways. However, there are certain traits of these definitions that overlap. Taking this overlap into consideration, a system can be defined as a typically hierarchical group of multiple parts that interact with each other to make a whole. Changes to these parts, therefore, change and evolve the system in its entirety.

A system consists of a group of multiple parts that interact with each other to make the whole. In Russell L. Ackoffs paper, "Systems thinking and thinking systems," Ackoff describes this trait of systems, stating that a system "cannot be divided into independent parts or subgroups of parts."[5] Because they cannot be divided into independent parts, they must, therefore, be dependent upon each other in order to be considered a part of a system. Furthermore, in Butler W. Lampsons paper, "Hints for Computer System Design," Lampson describes an issue that can come from a systems parts to interact within each other. He says that, since an interface, which is a system, " embodies assumptions which are shared by more than one part of a system, and sometimes by a great many parts, it is very desirable not to change the interface."[4] He claims that, through too much interaction and, therefore, reliance between parts of an interface, it causes difficulties when attempting to change the interface, since changing one part of the system may require changes to all the other parts that rely on it. In James Reasons paper, "Human error: models and management," Reason describes the implications of this iner-interactions of a system that makeup a whole. He claims that, "when an adverse event occurs" within a system, "the important issue is not who blundered,but how and why the defences failed."[3] What he is describing here is that a system is multiple parts, but it should be considered as a single thing. We shouldn't blame the single part that failed, but rather we should blame the interconnected parts as a whole that failed to prevent the failure.

A systems parts change and evolve the system as a whole. In Butler W. Lampsons paper, "Hints for Computer System Design," Lampson describes the difficulty when dealing with a complex system, stating that it can be"unclear about how one choice will limit freedom to make other choices, or affect the size and per- formance of the entire system."[4] Thus, Lampson claims that a choice can significantly affect the system as a whole, whether it be limiting freedom to make choices, the size of the system, or the performance of the system. Lampson goes on to describe how some programming languages mitigate these issues, claiming that with Mesa, a programming language that includes type-checking and support interfaces, it "becomes much easier to change interfaces without causing the system to collapse." [4]Furthermore, Lampson expands on methods to mitigate the effect that parts have on the system as a whole, stating that "it is easier to program and modify a system if its parts make fewer assumptions about each other."[4] In James Reasons paper, "Education and Debate", Reason claims that a single "strategic [decision]" made by a member of a system has "the potential for introducing pathogens into the system."[3] This result of a member of a systems decision highlights the fact that a systems parts change and evolve the entire system as a whole.

If complex, which systems most often are, systems are hierarchical with potentially many layers of

subsystems. In Lampsons paper, “Hints for Computer System Design”, he describes the difference between a computer system and designing an algorithm, stating that a computer system “has much more internal structure, and hence many internal interfaces.”[4] This idea of internal structure/interfaces is indicative of the hierarchical property of complex systems. Furthermore, in David Lorge Parnas paper, “Software Aspects of Defense systems”, Parnas describes what a proper structure of a system is, stating “The system should be divided into modules using information-hiding (abstraction) before writing the pro- gram begins.”[1] He claims that utilizing the hierarchical nature of systems to the fullest is beneficial in increasing the efficiency of dealing with them. In Reasons paper, “Education and Debate”, he states that “adherents of the system approach,” a hierarchical, systematic method, in life should seek “ a comprehensive management programme aimed at several different targets: the person, the team, the task, the workplace, and the institution as a whole” in order to effectively manage a system [3]. This approach manages a system by targeting the hierarchical nature of it. In Herbet A. Simons article, “The Architecture of Complexity.” he thoroughly depicts the hierarchical nature of complex systems, stating that a complex system consists of “ interrelated sub-systems, each of the latter being, in turn, hier-archic in structure until we reach some lowest level of elementary subsystem.” [2] He then goes on to give an example of a hierarchical complex system, stating that “business firms, governments, [and] universities all have a clearly visible parts-within-parts structure subordinated by an authority relation to the system it belongs.” [2]

In conclusion, based on the examples described previously taken from profound authors of various academic backgrounds and perspectives, a system can be defined as a typically hierarchical group of multiple parts that interact with each other to make a whole. Changes to these parts, therefore, change and evolve the system as a whole.

## References


[1] David Lorge Parnas. Software aspects of strategic defense systems. *Communications of the ACM*, 28(12):1326–1335, December 1985.

[2] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, 12 December 1962.

[3] James Reason. Human error: models and management. *BMJ*, 320:768–770, 2000.

[4] Butler W. Lampson. Hints for computer system design. *SIGOPS Operating Systems Review*, 17(5):3348, October 1983.

[5] Russell L. Ackoff. Systems thinking and thinking systems. *System Dynamics Review*, 10(2–3):175–188, Summer–Fall 1994.