Basic Software Architecture

A good understanding of the hierarchy of software currently employed by many modern digital switching systems is important. Most of today's digital switching systems employ quasi-distributed hardware and software architectures. The control structure of a digital switching system can usually be divided into three distinct levels. This chapter elaborates on the software employed in a hypothetical digital switching system at different levels of control. It shows levels of control along with some details of minimum software architectural functions that may be necessary for each level of control. From a digital switching system analyst's point of view, it is essential to understand the high level software architecture of a digital switch before attempting to analyze it. Low-level details are not essential, since the objective is to analyze the digital switch, not to design it. However, details on software engineering practices are essential for software analysis.

1. Operating Systems

Every digital switching system has an operating system as a part of its software architecture as shown in the fig 6.1 An operating system (OS) may be defined as software that manages the resources of a computer system or controls and tasks other programs. Sometimes these programs may be referred to as control programs, supervisory programs, executive programs, or monitor programs. In theory, there are different types of operating systems, classified as serial batch systems, multiprogramming systems, timesharing systems, and the real-time systems. The operating systems employed by digital switching systems are real-time operating systems. This type of OS is required for digital switching systems since the very nature of telephony processing demands execution of tasks in real time. Typically, the real-time operating system for the digital switching system interacts with different layers of applications necessary to support telephony features and functions. Since practically all modern digital switching systems use quasi-distributed architecture, the processor or controller for each subsystem may even use different OSs than the central processor does.
Therefore, it is conceivable for a digital switching system to employ more than one OS. Each subsystem may employ a different type of processor and therefore may employ different high-level languages for the development of its software. It is thus a challenge for the analyst to understand the operational and developmental environment of a digital switching system. Details on a methodology that could aid the analyst in better understanding the complex software environment of digital switching systems.

**Kernel.**

The kernel or the nucleus of an operating system comprises those functions of an OS that are most primitive to the environment. It usually supports the following functions.
- Process control and scheduling
- Main memory management
- Input/output control of requests for terminals and buffers
- Domain protection of main memory read/write operations etc.

Most of the real-time operating systems that control digital switching systems use priority interrupt systems. These interrupts are serviced by the kernel based on the importance of an operation. Of course, the interrupt and similar hierarchical controls are system-specific, but most
give highest priority to system maintenance interrupts, since this ensures proper operation of the
digital switching system, followed by other types of interrupts required for call processing and
other ancillary functions. Most digital switching systems employ kernels that reside in the main
memory.

2. Database Management
The databases that are employed in digital switching systems are usually
relational and sometimes distributed. In simple terms, distributed databases imply multiple
databases requiring data synchronization. The relational database systems use the relational data
model in which the relationships between files are represented by data values in file records
themselves rather than by physical pointers. A record in a relational database is flat, i.e., a
simple two dimensional arrangement of data elements. The grouping of related data items is
sometimes referred to as a tuple. A tuple containing two values is called a pair. A tuple
containing N values is called an TV-tuple. A good example of a relational database system in a
digital switching system is a database system that keeps cross-references of all directory
numbers that are assigned to the line equipment of subscribers. When a particular subscriber
goes off-hook, the line equipment is identified by the scanning program. The database is
searched to find its associated directory number that identifies all characteristics of the line. In
the hypothetical digital switching system developed for each network control processor (NCP) is
assigned a group of subscribers. Therefore, each NCP has a replica of the subscriber database
for all other NCPs. Depending on the type of call, a NCP may be required to route calls through
other NCPs. To accomplish this, the database information for all NCPs needs to be distributed
and always kept synchronized.

Concept of Generic Program
Most digital switching systems support the concept of generic program similar to release in the
computer industry. However, a generic program can be a little more involved than release. In the
early days of stored program control (SPC) switching systems, the generic program was more or
less the same for most telephone companies. Usually, the generic program contained all
programs necessary for the switching system to function. It included all switching software,
maintenance software, and specialized office data for the configuration of a central office (CO).
The translation data were usually supplied by the telephone companies. However, now it is
sometimes more difficult to define exactly what a generic program consists of. Most of the
modern digital systems have modular software structure. They usually have base programs or
core programs that control the basic functions of the digital switching systems. On top of these
programs reside different features and special options. Generally, the performance of a generic
program is tracked for software reliability. Therefore, it becomes very important for an analyst to identify the exact software components that constitute a generic program. The components of a digital switching systems software that are kept common for a specific market or a group of telephone companies can sometimes be used to identify the generic program. Usually this set of programs can be labeled as a generic, base, or core release for a digital switching system. In general, generic programs contain operating system(s), common switching software, system maintenance software, and common database(s) software for office data and translation data management.

**Software Architecture for Level 1 Control**

Level 1 is the lowest level of control. This level is usually associated with lines, trunks, or other low-level functions. Most of the software at this level is part of the switching software. As shown in Fig. 5.1, the interface controllers (ICs) are usually controlled by microprocessors and may have a small kernel controlling the hardware of the 1C. The ICs may have a small OS, labeled Operating System (Level 3) in Fig. 5.1. The function of this OS is to control and schedule all programs that are resident in the 1C. Most of the ICs have enough intelligence to recognize proper functioning of hardware and software. The 1C can also conduct diagnostics of lines and trunks or other peripherals connected to it. More extensive diagnostic routines may reside in the central processor or in some cases in the 1C itself. In either case/ the central processor can run the diagnostic program itself or request a fault-free 1C to run it. The 1C will then run the diagnostics and forward the results to the central processor. The ICs may also be capable of local recovery. This means that in case of an 1C failure, the 1C could recover itself without affecting the entire digital switching system. The only effect will be on the lines and trunk or peripherals connected to the 1C undergoing a recovery process. Again, all this will depend on the design of the ICs and associated software. An analyst should be conversant with different types of design strategies that may be employed, since they will impact the reliability and functionality of the 1C.

**Software Architecture for Level 2 Control**
The intermediate or level 2 controls are usually associated with network controllers that may contain distributed databases, customer data, and service circuit routines. Obviously these functions are digital switching architecture dependent; many switching functions could be assigned at this level of control. In a quasi-distributed environment, the processors employed are usually of intermediate or mini size. The NCPs are usually independent of the central processor. As shown in Fig. 5.1, the NCPs usually have their own operating system, labeled Operating System (Level 2). This OS has a kernel that controls the hardware and basic functionalities of the NCR. At this level of control, usually a resident database system maintains the translation data of subscribers and other software parameters required to control the telephony functions of the NCP. System recovery at this level of control is crucial, since a failure of a NCP may impact a number of ICs (dependent on the design) and a large number of lines, trunks, and peripherals. The NCPs should be capable of self-diagnosis, and since they are duplicated, they must be able to switch to a working backup. As mentioned in earlier chapters, the use of NCPs is design-specific. A design may call for a dedicated NCP to act as a control NCP for all other NCPs, or each NCP may be designed to operate independently. The recovery strategy in each case will be different. In the first case, where one NCP acts as the control NCP, the control NCP is responsible for system recovery for all other NCPs. In the second case, where there is no control NCP, the central processor is responsible for the recovery process of all NCPs. There could be all kinds of recovery strategies involved in the system recovery process at this level. The analyst needs to understand what type of recovery strategy is being used, in order to better assess the reliability of a digital switching system. Consider the function of the NCP. A subscriber goes off-hook, the IC receives an off-hook notification from the line module. The IC requests details on the subscriber, such as allowed features and applicable restrictions. The NCP queries its database for this information and passes it back to the IC. This type of action required by the NCP necessitates that the NCP maintain a subscriber database as well. This database is supposed to be managed and kept up to date with the latest information for each subscriber. This is shown as DBMS in Fig. 5.1 under level 2 control.

**Software Architecture for Level 3 Control**
The highest or level 3 control is usually associated with the central processor of a digital switching system. Normally these processors are mainframe type computers. Usually, the CP of a digital switching system provides all high level functions. These high-level functions include the management of the database system for office data, high-level subscriber data, software patch levels, feature control, and above all, system recovery in case of hardware or software failures. The main operating system of a modern digital switching system resides at this level and is labeled Operating System (Level 3) in Fig. 5.1. As mentioned earlier, this OS operates in real time and is multitasking (i.e., it can support more than one task at a time). This OS controls the database management system, switching software, recovery software, and all applications such as features, traffic management systems, and OS interfaces. Most CPs work in an active/standby mode. In this mode, one CP is always available to go into active mode if the active CP develops a fault. Indeed, there are different schemes for operating a redundant processor system to improve reliability and availability. However, for digital switching systems, the scheme most commonly employed is the one in which both processors execute instructions in a matched mode, and in case of a failure, the standby processor becomes active immediately. Other schemes are sometimes employed, such as hot standby, in which the standby processor is powered up and ready to take over the operation of an active processor. In this scheme, call processing can be impacted during the processor switchover. There is a third option, cold standby, in which the processor is not powered up, but can be brought on line in case of failure. This scheme is not used for CPs but is sometimes employed for less critical peripherals. Most of the maintenance and recovery functions of a switch are also controlled from this level.

**Digital Switching System Software Classification**

A conceptual diagram of typical digital switching system software is shown in Fig. 6.2. The basic software functionality of a digital switching system can be divided into five basic elements, and other functions can be derived from these basic elements:

- Switching software
- Maintenance software
- Office data
- Translation data

- Feature software

**Switching Software.**
The most important layer of software for a digital switching system usually comprises

- Call processing software

- Switching fabric control software

- Network control software

- Periphery control software

**Switch Maintenance Software.**
This set of programs is used to maintain digital switch software and hardware. Examples of these types of programs include digital switch diagnostics, automatic line tests, system recovery, patching, and trunk tests.

![Classification of digital switch software](image)

The recovery software of a modern digital switching system is usually distributed among its subsystems, since most digital switches have a quasi distributed architecture. This strategy allows the system to recover more efficiently. In earlier SPC systems, the recovery scheme required the entire switching system to go down before it could be reinitialized to a working configuration.
A digital switching system may employ a large number of programs that are external to the operation of the digital switch, such as operational support systems (OSSs), operator position support, and advanced features (e.g., ISDN/SCP AIN). These are not shown in Fig. 5.2 as separate items since they can be external to a digital switching system or may be implemented as a supported feature. Some parts of OSSs can even be viewed as 'part of digital switching system maintenance software.

The objective of Fig. 5.2 is to provide the analyst with a clear picture of the digital switch software. The objective of this chapter is to help the analyst better understand the software environment of a digital switch without getting distracted by functions that may not directly impact the reliability assessment of a class 5 digital switch. The importance of software tools such as compilers, assemblers, computer-aided software engineering tools, and methodologies that are needed to develop, produce, and maintain digital switching system software should not be ignored. They can impact the quality of software.

**Office Data.**

The generic program, as described earlier, requires information that is specific to a particular digital switch to operate properly. Digital switching systems have suffered outages due to wrong or improperly defined office data. The easiest way to visualize office data is by comparing them to your personal computer (PC). For the PC to operate properly, the OS has to know what type of color monitor the PC is equipped with, so that correct drivers are installed; the size and type of hard disk installed so that it can access it correctly; types of floppy disks/mouse; and CD ROM. Similarly, the office data of a digital switching system describe the extent of a central office (CO) to the generic program. However, the office data are much more involved and also define software parameters along with hardware equipment. Some common hardware parameters are

- Number of NCP pairs in the CO
- Number of line controllers in the CO
- Maximum number of lines for which the CO is engineered
- Total number of line equipment in the CO
- Maximum number of trunks and types of trunks for which the CO is engineered
- Total number of trunks of each type in the CO

- Total number and types of service circuits in the CQ such as ringing units, multifrequency (MF) receivers and transmitters, and dial-pulse (DP) receivers and transmitters. These are some examples of software parameters:
- Size of automatic message accounting (AMA) registers

- Number of AMA registers

- Number and types of traffic registers

- Size of buffers for various telephony functions

- Names and types of features supported

These types of parameters are digital switching system-specific and CO-specific. The parameters can literally number in the hundreds and are generated from engineering specifications of a CO.

**Translation Data.**

The translation data, also referred to as subscriber data, are subscriber-specific and are required for each subscriber. This type of data is generally generated by the telephone companies and not by the suppliers. In some cases, the suppliers may input translation data supplied by the telephone companies. However, the database and entry system for the translation data is supplied as part of the digital switching system software. Typical translation data may consist of:

- Assignment of directory number to a line equipment number

- Features subscribed to by a particular customer, such as call waiting, three-way calling, and call forwarding, etc.

- Restrictions for a particular customer, such as incoming calls only, no long-distance calls, certain calls blocked

- Three-digit translators that route the call based on the first three digits dialed
- Area-code translators that translate the call to a tandem office for 1+ call, which is followed by 10 digits

- International call translators that route the call to international gateway offices based on the country code dialed

Again, literally hundreds of translation tables are built for a CO before it can become functional. If the CO is a new installation, much of the information is provided by the traffic department of a telephone company. The data tables are generated in conjunction with the specification of a new CO. However, if the CO is a replacement for an earlier CO then all existing data may be required to be regenerated in a different format for the new CO.

**Feature Software.**

As mentioned earlier, most features implemented in modern digital switching systems are offered through feature packages. Some of the feature packages are put in a feature group and are offered in a certain market or to a group of telephone companies. These features may be included in the base package of a generic release or, offered as an optional package. In either case, most of the features are considered to be applications for a digital switch. They are engineered to be modular and can be added to a digital switch according to the requirements of the telephone company and associated CO Some examples of feature packages are

- Operator services b Centrex feature

- ISDN basic rate « STP extensions

- SCP database

Depending on the digital switching system, these feature packages can be extensive and large. The analyst of digital switch software should assess the extent of the feature package and its compliance with the requirements of telephone companies.

**Software Dependencies.**

Most telephony features of digital switching systems require specific office data and translation data for their operation. They depend on the generation of feature specific office data and/or translation data. These dependencies are, of course, design-specific. Similarly, the maintenance programs may require a set of specialized office data and/or translation data for testing various
functionalities of a digital switching system. These relationships are shown as a software dependency in Fig. 5.2, and direct interactions of a generic program are shown as solid arrows.