Debugging & Traceability

Date	16 November 2022
Team ID	PNT2022TMID50111
Project	IoT Based Smart Crop Protection System For Agriculture

In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing, and removing errors. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

Debugging Process: Steps involved in debugging are:

Problem identification and report preparation.
Assigning the report to the software engineer to the defect to verify that it is genuine. Defect Analysis using modelling , documentation, finding and testing candidate flaws, etc. Defect Resolution by making required changes to the system. Validation of corrections.

The debugging process will always have one of two outcomes:

1. The cause will be found and corrected.

2. The cause will not be found.

Later, the person performing debugging may suspect a cause, design a test case to help validate that suspicion and work toward error correction in an iterative fashion.

During debugging, we encounter errors that range from mildly annoying to catastrophic. As the consequences of an error increase, the amount of pressure to find the cause also increases. Often, pressure sometimes forces a software developer to fix one error and at the same time introduce two more.

Debugging Approaches/Strategies:

1. **Brute Force:** Study the system for a larger duration in order to understand the system. It helps the debugger to construct different representations of systems to be debugging

depending on the need. A study of the system is also done actively to find recent changes made to the software.

2. **Backtracking:** Backward analysis of the problem which involves tracing the program backward from the location of the failure message in order to identify the region of faulty code. A detailed study of the region is conducted to find the cause of defects.
3. **Forward analysis** of the program involves tracing the program forwards using breakpoints or print statements at different points in the program and studying the results. The region where the wrong outputs are obtained is the region that needs to be focused on to find the defect.
4. **Using the past experience** of the software debug the software with similar problems in nature. The success of this approach depends on the expertise of the debugger.
5. **Cause elimination:** it introduces the concept of binary partitioning. Data related to the error occurrence are organized to isolate potential causes.

Debugging Tools:

Debugging tool is a computer program that is used to test and debug other programs. A lot of public domain software like gd band dbx are available for debugging. They offer console-based command-line interfaces. Examples of automated debugging tools include code based tracers, profilers, interpreters, etc. Some of the widely used debuggers are:

Radare2
WinDbg
Valgrind

Difference Between Debugging and Testing:

Debugging is different from testing. Testing focuses on finding bugs, errors, etc whereas debugging starts after a bug has been identified in the software. Testing is used to ensure that the program is correct and it was supposed to do with a certain minimum success rate. Testing can be manual or automated. There are several different types of testing like unit testing, integration testing, alpha and beta testing, etc. Debugging requires a lot of knowledge, skills, and expertise. It can be supported by some automated tools available but is more of a manual process as every bug is different and requires a different technique, unlike a pre-defined testing mechanism.

Debugging and Tracing

Last Updated: 2021-03-01

This information describes the tracing and debugging facilities that are available with SA z/OS.

To collect debugging information you can also use the command INGLKUP REQ=COLLECT.

Automation Manager State Trace Table
: The event handler trace back table is written to SYSLOG or to the Message Logger. It is the most important debug tool when you want to understand the event flow.

Using Trace Services for the Automation Manager
: To trace the control flow of a process, SA z/OS uses the **MVS™ Component TraceFacility** with all its capabilities.

Using AOCTRACE
: You can use the AOCTRACE command to enable or disable the automation debugging facility, either globally or for specific lists (REXX routines).

How to Use a Log with INGHIST
: With the INGHIST command you can display a log with automation manager messages by entering `INGHIST REQ=LOG` at the command line.

How to Use the Diagnostic Option of INGAMS
: From the INGAMS command dialog, you can select option **D Diagnostic**.

Tracing and Debugging for Proc Ops and the BCP Internal Interface
: The following trace facilities are provided for Proc Ops SNMP connections and the BCP Internal Interface that you can use for problem determination purposes.

Using Trace Services for I/O Operations
: To trace the control flow of I/O operations, SA z/OS uses the MVS Component Trace Facility with all its capabilities.

Collecting the available BCPII session status messages
: In case of BCP ii session problems, make sure you have collected all SA z/OS WTO messages (msg ids ING81*) at the time of the error, that are issued in the SA z/OS Net View domain, where the error occurred. Report these messages (Sys log, Net log) in case you need to contact IBM® Support.