COMP 3550

9.2 — BUILDING & TRACING THROUGH LEGACY SYSTEMS

Week 9: Legacy Software, Architecture Recovery & Change

FIRST STEP — BUILD THE PROJECT

Why Start Here?

- You can’t safely change what you can’t run
- Building confirms you have the right environment & dependencies
- A working baseline is your “safety net” for future changes

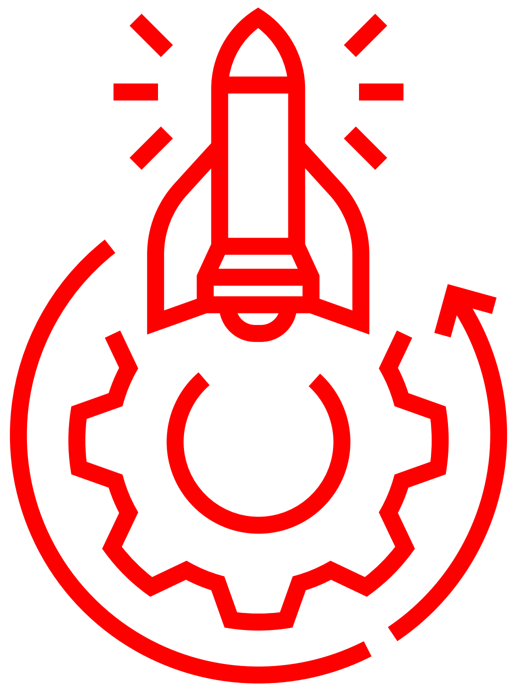
FIRST STEP — BUILD THE PROJECT

Why Start Here?

- You can’t safely change what you can’t run
- Building confirms you have the right environment & dependencies
- A working baseline is your “safety net” for future changes

Confirm the Project Compiles / Runs

- Install required SDKs, frameworks, libraries
- Resolve missing dependencies or environment variables
- Document build steps for your future self



FIRST STEP — BUILD THE PROJECT

Why Start Here?

- You can’t safely change what you can’t run
- Building confirms you have the right environment & dependencies
- A working baseline is your “safety net” for future changes

Confirm the Project Compiles / Runs

- Install required SDKs, frameworks, libraries
- Resolve missing dependencies or environment variables
- Document build steps for your future self

Set Up Debugging

- Use IDE tools to configure breakpoints & watch variables
- Verify you can step through code line-by-line
- Confirm logs/console output are visible

RUNTIME TRACING TECHNIQUES

Understand how the system behaves in real time — not just what the code looks like.

Add Breakpoints

- Pause execution at key lines or method entries
- Inspect variable values and call stack
- Step into / over / out to control the flow

RUNTIME TRACING TECHNIQUES

Understand how the system behaves in real time — not just what the code looks like.

Add Breakpoints

- Pause execution at key lines or method entries
- Inspect variable values and call stack
- Step into / over / out to control the flow

Use Logging

- Insert temporary log statements for key events
- Capture variable values, timestamps, and branch choices
- Keep logs lightweight to avoid noise

RUNTIME TRACING TECHNIQUES

Understand how the system behaves in real time — not just what the code looks like.

Add Breakpoints

- Pause execution at key lines or method entries
- Inspect variable values and call stack
- Step into / over / out to control the flow

Use Logging

- Insert temporary log statements for key events
- Capture variable values, timestamps, and branch choices
- Keep logs lightweight to avoid noise

Follow a Main Scenario

- Pick a common user flow (e.g., Login → Dashboard)
- Trace the execution path through controllers, services, and data layers
- Note dependencies and side effects

THE “SCRATCHPAD METHOD”

- **Track Inputs / Outputs of Key Methods**
  - Note what goes in and what comes out
  - Record changes to important objects or variables
  - Identify where data originates and where it flows
- **Build a Rough Diagram as You Go**
  - Boxes for components (classes, services, modules)
  - Arrows for data flow or method calls
  - No need for perfection, this is for you, not a formal doc (a bit like field notes)

TOOLS TO HELP YOU TRACE

IDE & Code Navigation Tools

- Find Usages (a.k.a. Find References)
- See where a class, method, or variable is used
- Great for spotting entry points & ripple effects

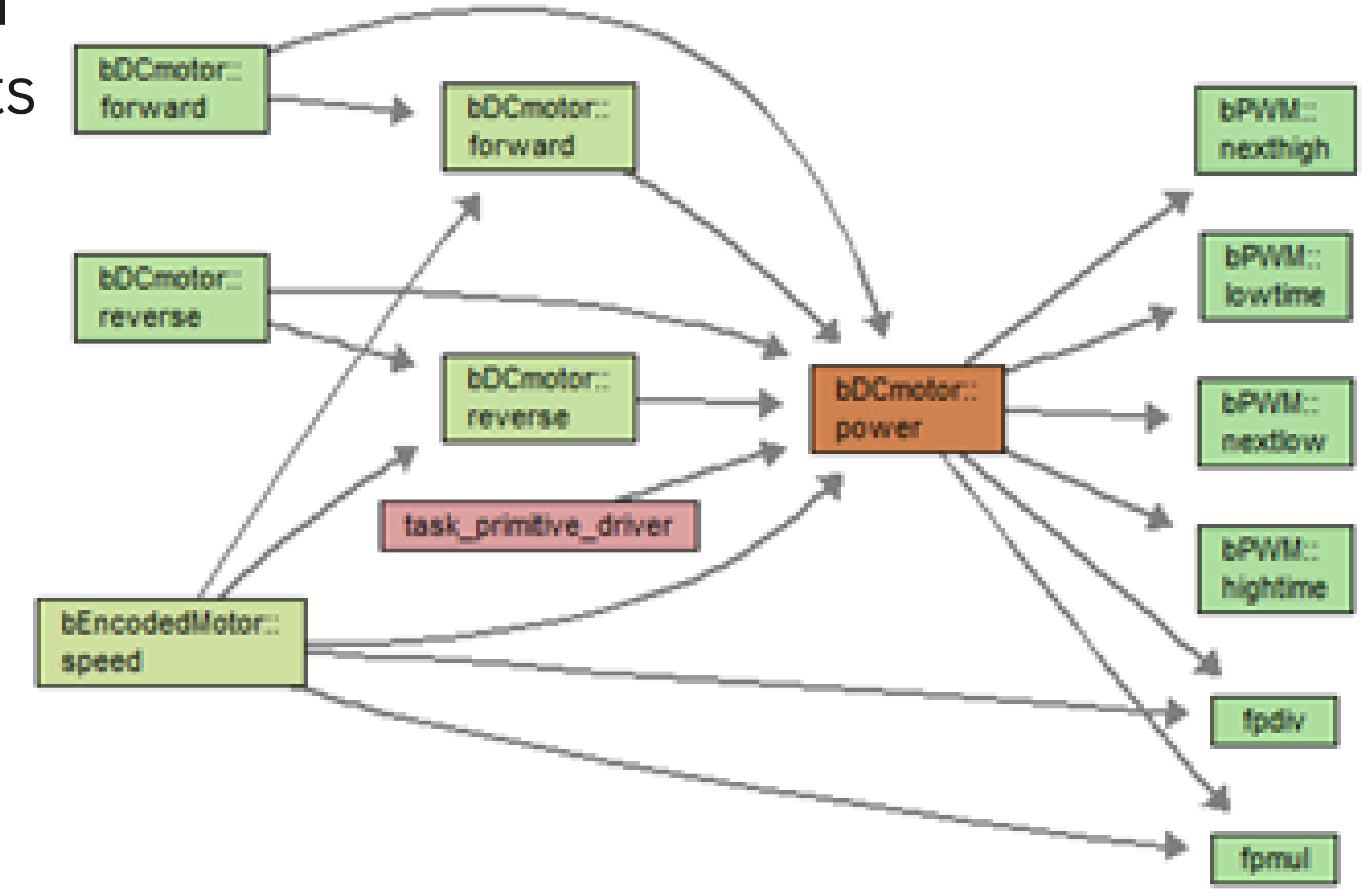
TOOLS TO HELP YOU TRACE

IDE & Code Navigation Tools

- Find Usages (a.k.a. Find References)
- See where a class, method, or variable is used
- Great for spotting entry points & ripple effects

Call Graphs

- Visualize which methods call which others
- Helps identify critical paths and dead ends



https://www.imagix.com/appnotes/function-calls-graph.html

TOOLS TO HELP YOU TRACE

IDE & Code Navigation Tools

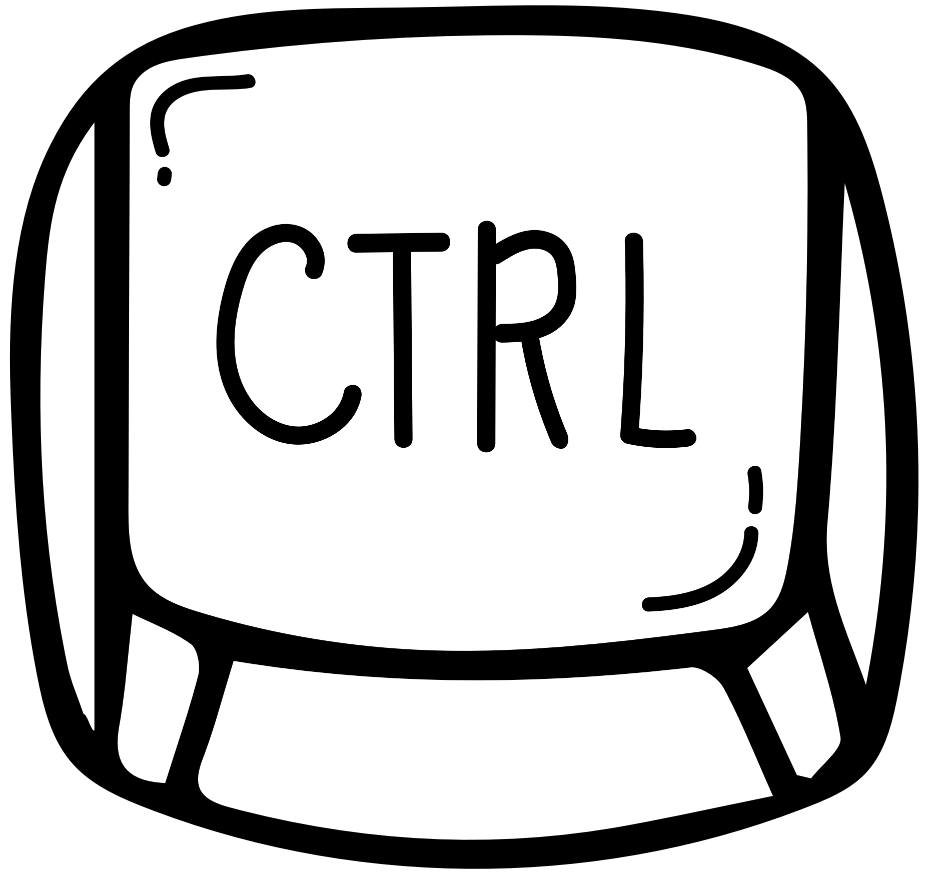
- Find Usages (a.k.a. Find References)
- See where a class, method, or variable is used
- Great for spotting entry points & ripple effects

Call Graphs

- Visualize which methods call which others
- Helps identify critical paths and dead ends

Ctrl+Click (Go to Definition)

- Jump instantly to a method, class, or variable definition
- Follow the code flow without losing your place



PROJECT PAUSE & REFLECT

Open a random method in your project.
Can you trace it to a feature or UI element?

What about in another group’s project?

Try sketching out a call graph for the last method you or a teammate made.