FORMALNE METODE

- matematički zasnovane tehnike za specificiranje zahtjeva i arhitekture te verifikaciju sustava
- doprinose pouzdanosti i robustnosti, statička verifikacija, dopuna testiranju (dinamičkoj)
- zajedničko svim FM - jezik (način zapisa + semantika), alat (traži sve slučajeve), metodologija
- primjena: u životnom ciklusu sustava, na komponente sustava, na svojstva komponente
- za sustave: kritično važne, život kritične,informacijske i poslovne s visokom razinom sigurnosti
- bit svega: ispravnost
- vrste:

1. **FORMALNA SPECIFIKACIJA**

- matematički model visoke razine apstrakcije, prokazuje nekonzistentne i dvosmislene specifikacije, sam je tehnika, nema dodatnih alata

- **ASM metoda**
    - 3 temelja: ASM stroj (FSM) generaliziranih stanja, ASM temeljni model (matematički opis), ASM rafiniranje (povezivanje modela dobivenih sukcesivnim koracima specifikacije)
    - Konačan skup if-then pravila, stanje -> odabir pravila, pridruživanje vrijednosti, izračunavanje novih stanja
- **Z metoda (“zed”)**
    - Z dokument - matematičke definicije + engleski tekst, Z tekst pisan kao skup struktura - Schema
    - Opis definicija i ograničenja, zatim akcija, završava dokazivanjem teorema
    - Tipovi schema: state schema (globalne izjave), operation schema (opis operacija koje mijenjaju stanje), observation schema (opisuje dohvat informacija)

2. **FORMALNA SINTEZA**

- Oblikovanje sustava odozgo prema dolje (apstracija -> implementacija)

- **SDL (Specification and Description Language)**
    - Za specifikaciju i sintezu sustava za rad u stvarnom vremenu i arhitekturi programskih proizvoda temeljenoj na implicitnom pozivanju (event based)
    - Dokazano formalno potpun, podržava analizu specifikacija s aspekta potpunosti, ispravnosti i konzistencije
    - Integracija grafičkog jezika (SDL/GR) i tekstualnog jezika temeljenog na simbolima (SDL/PR - phrase representation)
    - Strukturiran u hijerarhijske razine:
        - Sutruktura sustava: sustav, blokovi
        - Ponašanje sustava: procesi, procedure
    - Razmjenjivanje informacija porukama
- **Prevoditelj (compiler)**
    - Izvorni -> ciljni jezik, za generiranje izvršnog koda treba druge programe
    - Interpreter samo izvodi programe
    - Izvodni prog => compiler -> ciljni asemblerski prog => asember -> relokatibilni strojni kod => linker -> izvršni kod => loader -> razrješenje adresa i punjenje
    - Arhitektura:
        - Analiza: leksička, sintaksna, semantička

    - Sinteza: gener. prijelaznog koda, optimizacija koda, gen. ciljnog koda

3. FORMALNA VERIFIKACIJA

- Debugging - znamo da bug postoji, ali mu ne znamo mjesto
- FV: dokazivanje postojanja ili nepostojanja buga
- Najčešće nad modelima stvarnih programa
- Metode:
    - **Provjera modela** - zadovoljava li sustav obilježje
        - Automatizirana provjera reaktivnih sustava uz pomoć FSA
        - Pretraživanje prostora stanja (baš svih)
    - **Dokazivanje teorema** - je li specif. log. posljedica implementacije
        - Ulazi: logičke formule implementacije (aksiomi), log forumula specifikacije, pretpostavke o domeni, teorija
        - Izlaz: implementacija I= specifikacija
    - **Provjera ekvivalentnosti** - usporedba dvaju modela/implementacija
        - Kombinacijska provjera - funkcijska ekvivalentnost sustava bez memorije (stanja) - uspoređivanje izlaza za iste ulaze
        - Sekvencijska provjera - ekvivalencija za svako valjano stanje sustava, stvaranje zajedničkog FSM-a i ulaznog vektora **x**, moraju dati isti izlaz za svako dosezljivo stanje
    - **Provjera tvrdnje (assertion)** - usporedba specifičnog svojstva
        - Opisivanje očekivanog ili neočekivanog uvjeta
        - DUT - device under test
- Podjela po težini:
    - **Lagane** - neispravne i nekompletne, naivno pretraživanje - provj modela
    - **Srednje teške** - ispravne, nekompletne, preko čvrste točke - provj tvrdnje
    - **Teške** - ispravne i kompletne - složena obilježja, mat dokazi, veliki resursni zahtjevi - dokazivanje teorema