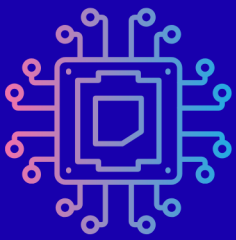
2026 Edition

The Future Coder Series

Computer Science စာအုပ်

Vol 2: The Logic and Data



ရန်နိုင်လင်း (Software Engineer)

MBA, PG Dip (CS), BSc(Physics)

Chapter 1: Systems & Information

1.1 Data နှင့် Information ကွာခြားချက်

1.2 System (စနစ်) ဆိုတာ ဘာလဲ?

1.3 Information System (IS) ၏ အစိတ်အပိုင်းများ

1.4 အရည်အသွေးရှိသော Information ၏ ဂုဏ်သတ္တိများ

Chapter 2: Algorithms & Problem Solving

2.1 Algorithm (အယ်လဂိုရီသမ်) ဆိုတာ ဘာလဲ?

2.2 Program Design Tools (ဒီဇိုင်းရေးဆွဲသည့် ကိရိယာများ)

2.3 Structured Programming (စနစ်တကျ တည်ဆောက်ပုံ ၃ မျိုး)

2.4 Modular Design (အစိတ်အပိုင်းငယ်များ ခွဲခြင်း)

Chapter 3: Software Development Life Cycle (SDLC)

3.1 SDLC ဆိုတာ ဘာလဲ?

3.2 Waterfall Model

3.3 Agile Model (ခေတ်သစ် ပေါ့ပါးသွက်လက်သော ပုံစံ)

3.4 Prototyping (နမူနာပြ ပုံစံ)

Chapter 4: System Modeling

4.1 System Modeling ဆိုတာ ဘာလဲ?

4.2 Data Flow Diagram - DFD (အချက်အလက် စီးဆင်းပုံပြ ဇယား)

4.3 Decision Tables (ဆုံးဖြတ်ချက် ဇယားများ)

4.4 Unified Modeling Language - UML (ခေတ်သစ် စံပြုပုံစံ)

Chapter 5: Data Modeling & Databases

5.1 Data Modeling ဆိုတာ ဘာလဲ?

5.2 Entity-Relationship Diagram (ERD)

5.3 Normalization (အချက်အလက်များကို သန့်စင်ခြင်း)

5.4 File Organization & Database (ဖိုင်မှ ဒေတာဘေ့စ်သို့)

Chapter 1: Systems & Information

(စနစ်များနှင့် သတင်းအချက်အလက် နည်းပညာ)

1.1 Data နှင့် Information ကွာခြားချက်

ကွန်ပျူတာပညာရှင်တစ်ယောက် ဖြစ်လာဖို့အတွက် ပထမဆုံး နားလည်ထားရ မယ့် အယူအဆကတော့ Data နဲ့ Information မတူဘူး ဆိုတာပါပဲ ။

(a) Data (အကြမ်းထည် အချက်အလက်)

Data ဆိုတာ အလုပ်မလုပ်ရသေးတဲ့၊ အဓိပ္ပာယ် မဖော်ရသေး တဲ့ အချက်အလက် အကြမ်းထည်တွေ ဖြစ်ပါတယ် ။

- **ဥပမာ:** 100, Red, Mg Mg။
- ဒီအတိုင်းကြည့်ရင် 100 ဆိုတာ ပိုက်ဆံလား၊ အမှတ်လား၊ အသက် လား မသိနိုင်ပါဘူး။ ဒါဟာ စာရွက်ပေါ်က ဂဏန်းအဆင့်သာ ရှိပါသေး တယ်။

(b) Information (သတင်းအချက်အလက်)

Information ဆိုတာ Data ကို အသုံးဝင်အောင် စီမံထားတဲ့ ရလဒ်ဖြစ်ပါတယ် ။ Data ကို ဆုံးဖြတ်ချက်ချဖို့ (Decision Making) အသုံးဝင်အောင် ပြောင်းလဲ လိုက်တဲ့အခါ Information ဖြစ်လာပါတယ် ။

ဥပမာ (E-commerce):

- **Data:** ဝယ်သူတစ်ယောက်က ခေါက်ဆွဲခြောက် ၁၀ ထုပ် ဝယ်သွား တယ်။ (ဒါက ရိုးရိုး အရောင်းစာရင်းပါ)။
- **Information:** "ဒီဝယ်သူက လကုန်ရက်တိုင်း ခေါက်ဆွဲခြောက် ဝယ်လေ့ ရှိတယ်" ဆိုပြီး ကွန်ပျူတာက သုံးသပ်ပြလိုက်ရင် ဒါ ဟာ Information ဖြစ်သွားပါပြီ။

- **Knowledge (ဗဟုသုတ – Modern Addition):** "လကုန်ရက်တွေမှာ ခေါက်ဆွဲခြောက်တွေ ပိုတင်ထားသင့်တယ်" ဆိုပြီး ဆုံးဖြတ်ချက်ချနိုင်တဲ့ အဆင့်ကို ဆိုလိုပါတယ်။

1.2 System (စနစ်) ဆိုတာ ဘာလဲ?

System ဆိုတာ ဘုံရည်မှန်းချက်တစ်ခု (Common Goal) ရရှိဖို့ အတွက် အစိတ်အပိုင်းတွေ ပေါင်းစပ်အလုပ်လုပ်နေတာကို ခေါ်ပါတယ် ။ စနစ်တစ်ခုမှာ အဓိက အစိတ်အပိုင်း (၃) ခု ရှိပါတယ်:

1. **Input (အဝင်):** ကုန်ကြမ်းတွေ၊ Data တွေ ထည့်သွင်းခြင်း ။
2. **Processing (လုပ်ဆောင်ခြင်း):** Input ကို လိုချင်တဲ့ပုံစံပြောင်းလဲခြင်း ။
3. **Output (အထွက်):** ရလာတဲ့ ရလဒ်ကို ထုတ်ပြခြင်း ။
4. **Feedback (တုံ့ပြန်မှု):** Output က မှန်ရဲ့လားဆိုတာ ပြန်စစ်ဆေးပြီး Input ကို ပြုပြင်ခြင်း ။

ဥပမာ – Taxi System:

- **Input:** ခရီးသည်က လက်ရှိနေရာနဲ့ သွားချင်တဲ့နေရာ ရိုက်ထည့်လိုက်တယ်။
- **Process:** System က အနီးဆုံးကားကို ရှာတယ်၊ ဈေးနှုန်းတွက်တယ်။
- **Output:** ကားဆရာဆီကို အကြောင်းကြားစာ ရောက်သွားတယ်။
- **Feedback:** ခရီးစဉ်ပြီးဆုံးကြောင်း ကားဆရာက အကြောင်းပြန်ရင် System က ငွေရှင်းပေးလိုက်တယ်။

1.3 Information System (IS) ၏ အစိတ်အပိုင်းများ

Information System (သတင်းအချက်အလက်စနစ်) ဆိုတာ လူတွေ၊ ဟာ့ဒ်ဝဲ တွေ၊ ဆော့ဖ်ဝဲတွေ ပေါင်းစပ်ပြီး အလုပ်လုပ်တဲ့ စနစ်ကြီးဖြစ်ပါတယ် ။
အဓိက အစိတ်အပိုင်း (၅) ခု ပါဝင်ပါတယ်:

(a) People (လူများ)

စနစ်ကို သုံးစွဲသူ (End Users) နဲ့ စနစ်
ကို တည်ဆောက်သူ (System Analysts, Programmers) တွေ ဖြစ်ပါတယ် ။ လူ မရှိရင် စနစ်က အသက်မဝင်ပါဘူး ။

(b) Hardware (စက်ပစ္စည်းများ)

ကွန်ပျူတာများ၊ ဖုန်းများ၊ Server ကြီးများ၊ Barcode Scanner များ ပါဝင်ပါ တယ် ။

(c) Software (ပရိုဂရမ်များ)

- **System Software:** Windows, Linux, Android (Operating Systems) ။
- **Application Software:** Excel, Point of Sales (POS) software, Banking App ။

(d) Data (အချက်အလက်များ)

Text တွေ၊ ဓာတ်ပုံတွေ၊ အသံဖိုင်တွေ အားလုံးပါဝင်ပါတယ် ။ ခေတ်သစ်စနစ် တွေမှာတော့ ဒါတွေကို **Database** (အချက်အလက်တိုက်) တွေထဲ မှာ သိမ်းဆည်းကြပါတယ်။

(e) Network & Procedures (ကွန်ရက်နှင့် လုပ်ထုံးလုပ်နည်းများ)

- **Network:** အင်တာနက်၊ WiFi၊ Fiberလိုင်းများ ။

- **Procedures:** စနစ်ကို ဘယ်လိုသုံးရမလဲဆိုတဲ့ လမ်းညွှန်ချက်များ (Manuals) ။

1.4 အရည်အသွေးရှိသော Information ၏ ဂုဏ်သတ္တိများ

Information တစ်ခုဟာ ဆုံးဖြတ်ချက်ချဖို့ ကောင်းမွန်နေရပါမယ်။
ကောင်းမွန်သော Information တစ်ခုမှာ အောက်ပါအချက်များ ပြည့်စုံရပါမယ်:

- **Accurate (မှန်ကန်မှု):** အမှားကင်းရမယ် ။
- **Timely (အချိန်မီမှု):** လိုအပ်တဲ့အချိန်မှာ ရရှိရမယ်။
- **Relevant (ဆီလျော်မှု):** ကိုယ်လုပ်မယ့်အလုပ်နဲ့ သက်ဆိုင်နေရမယ် ။
- **Complete (ပြည့်စုံမှု):** လိုအပ်တဲ့ အချက်အလက် အားလုံးပါရမယ် ။

Chapter 2: Algorithms & Problem Solving

(ပြဿနာဖြေရှင်းနည်းများနှင့် လုပ်ငန်းစဉ်များ)

2.1 Algorithm (အယ်လဂိုရီသမ်) ဆိုတာ ဘာလဲ?

Algorithm ဆိုတာ ပြဿနာတစ်ခုကို ဖြေရှင်းဖို့ သို့မဟုတ် အလုပ်တစ်ခု ပြီးမြောက်ဖို့အတွက် တစ်ဆင့်ပြီးတစ်ဆင့် လုပ်ဆောင်ရ

မယ့် **"တိကျသော ညွှန်ကြားချက်များ" (Step-by-step Instructions)** ဖြစ်ပါတယ်။

ဟင်းချက်နည်း (Recipe) ဥပမာ: ကြက်ဥကြော်ဖို့အတွက် -

1. ဒယ်အိုးကို မီးဖိုပေါ်တင်ပါ။
2. ဆီထည့်ပြီး ပူလာအောင် စောင့်ပါ။
3. ကြက်ဥဖောက်ထည့်ပါ။
4. ကျက်ရင် ပန်းကန်ထဲ ထည့်ပါ။

ဒီအဆင့်တွေကို Algorithm လို့ခေါ်ပါတယ်။ ကွန်ပျူတာ ပရိုဂရမ်ဆိုတာ တကယ်တော့ Algorithm တွေကို ကွန်ပျူတာနားလည်တဲ့ ဘာသာစကား (Code) နဲ့ ပြောင်းရေးထားခြင်းသာ ဖြစ်ပါတယ်။

2.2 Program Design Tools (ဒီဇိုင်းရေးဆွဲသည့် ကိရိယာများ)

အိမ်မဆောက်ခင် ပုံကြမ်း (Blueprint) ဆွဲရသလိုပဲ၊ ကုဒ်မရေးခင်မှာ Algorithm ကို အရင်ပုံဖော်ရပါတယ်။ အဓိက နည်းလမ်း (၂) ခု ရှိပါတယ် -

(a) Pseudocode (ဆူဒိုကုဒ် – ကုဒ်အတု)

ဒါက တကယ့် Programming Language (ဥပမာ Python) မဟုတ်ဘဲ၊ လူနားလည်တဲ့ အင်္ဂလိပ်စာ (သို့မဟုတ် မြန်မာစာ) နဲ့ Logic ကို ရေးသားထားခြင်း ဖြစ်ပါတယ် ။

- **စည်းကမ်း:** သတ်မှတ်ထားတဲ့ Grammar မရှိပါဘူး။ ဖတ်လိုက်တာနဲ့ နားလည်ရင် ရပါပြီ။
- **Modern Style:** ယနေ့ခေတ်မှာ Python ကုဒ်နဲ့ ဆင်တူတဲ့ ပုံစံမျိုး ရေးလေ့ရှိပါတယ်။

ဥပမာ – စာမေးပွဲ အောင်/ရှုံး စစ်ဆေးခြင်း:

```
INPUT Score
IF Score >= 40 THEN
   PRINT "Passed (အောင်မြင်သည်)"
ELSE
   PRINT "Failed (ကျရှုံးသည်)"
ENDIF
```

(b) Flowchart (စီးဆင်းမှုပုံပြ ဇယား)

Algorithm ရဲ့ အဆင့်တွေကို ပုံတွေ၊ မြှားတွေနဲ့ ဆက်စပ်ပြသခြင်း ဖြစ်ပါတယ် ။
Flowchart ဆွဲရာမှာ နိုင်ငံတကာ စံသတ်မှတ်ထား
တဲ့ သင်္ကေတ (Symbols) တွေကို သုံးရပါတယ် ။

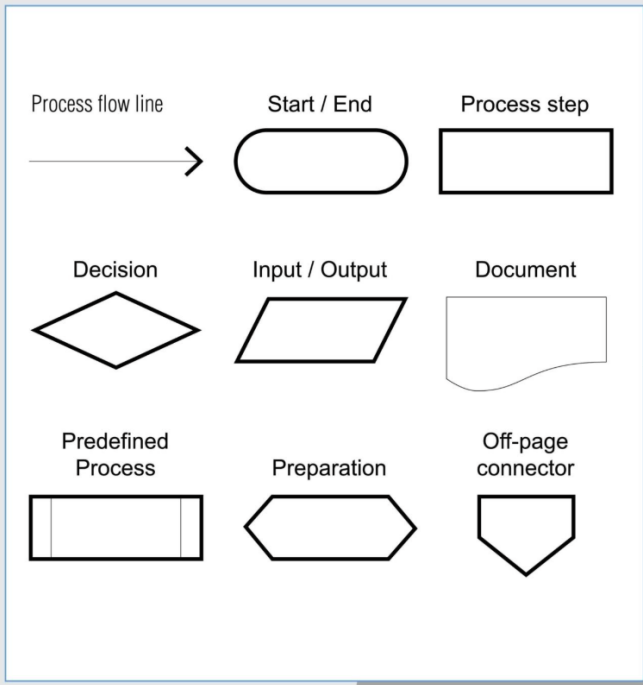


image - Shutterstock

- **Oval (ဘဲဥပုံ):** Start / End (အစ နှင့် အဆုံး)။
- **Parallelogram (အနားမြိုင်စတုဂံပုံ):** Input / Output (အချက်အလက် သွင်းခြင်း/ထုတ်ပြခြင်း)။
- **Rectangle (လေးထောင့်ပုံ):** Process (တွက်ချက်ခြင်း၊ လုပ်ဆောင်ခြင်း)။

- **Diamond (စိန်ပွင့်ပုံ):** Decision (ဟုတ်/မဟုတ် ဆုံးဖြတ်ခြင်း)။
- **Arrows (မြှားများ):** Flowlines (လုပ်ငန်းစဉ် စီးဆင်းရာ လမ်းကြောင်း)။

2.3 Structured Programming (စနစ်တကျ တည်ဆောက်ပုံ ၃ မျိုး)

မည်မျှ ရှုပ်ထွေးသော ပရိုဂရမ်ဖြစ်ပါစေ၊ အောက်ပါ အခြေခံ တည်ဆောက်ပုံ (Control Structures) ၃ မျိုးနဲ့ပဲ ဖွဲ့စည်းထားပါတယ် ။

(a) Sequence (အစဉ်လိုက် လုပ်ဆောင်ခြင်း)

အဆင့်တွေကို တစ်ခုပြီးမှ တစ်ခု အစဉ်လိုက် လုပ်ဆောင်ခြင်း ဖြစ်ပါတယ် ။

- **Logic:** Step A -> Step B -> Step C
- **Example:**
  - Input A
  - Input B
  - Sum = A + B
  - Print Sum

(b) Selection (ရွေးချယ် ဆုံးဖြတ်ခြင်း)

အခြေအနေတစ်ခုပေါ် မူတည်ပြီး လမ်းကြောင်းခွဲလိုက်ခြင်း ဖြစ်ပါတယ် (Decision Making) ။

- **Logic:** IF (Condition is True) THEN (Do This) ELSE (Do That).
- **Flowchart:** Diamond ပုံကနေ Yes လမ်းကြောင်းနဲ့ No လမ်းကြောင်း ၂ ခု ကွဲထွက်သွားပါမယ်။

(c) Repetition / Iteration (ထပ်ခါတလဲလဲ လုပ်ဆောင်ခြင်း)

သတ်မှတ်ထားတဲ့ အခြေအနေတစ်ခု ပြည့်မီနေသမျှ ကာလပတ်လုံး အလုပ်တစ်ခုကို ထပ်ခါထပ်ခါ လုပ်နေခြင်း (Looping) ဖြစ်ပါတယ် ။

- **While Loop:** အခြေအနေ မှန်နေသမျှ လုပ်မယ်။
  - ဥပမာ: While (Battery < 100%), Charge Phone. (ဘက်ထရီ မပြည့်မချင်း အားသွင်းနေမယ်)။
- **For Loop:** သတ်မှတ်ထားတဲ့ အရေအတွက်အတိုင်း လုပ်မယ်။
  - ဥပမာ: For i = 1 to 10, Print i. (၁ ကနေ ၁၀ အထိ ဂဏန်းထုတ်မယ်)။

2.4 Modular Design (အစိတ်အပိုင်းငယ်များ ခွဲခြင်း)

ပြဿနာအကြီးကြီးတွေကို ဒီအတိုင်းဖြေရှင်းမယ့်အစား၊ အပိုင်းငယ်လေးတွေ (Modules/Sub-programs) ခွဲပြီး ဖြေရှင်းတာ ပိုထိရောက်ပါတယ် ။

Top–Down Approach:

Main Problem: ကျောင်းသား စီမံခန့်ခွဲမှု စနစ်။

- **Module 1:** ကျောင်းသားစာရင်းသွင်းခြင်း။
- **Module 2:** အမှတ်စာရင်း တွက်ချက်ခြင်း။
- **Module 3:** လစဉ်ကြေး ပေးသွင်းခြင်း။

ဒီလိုခွဲလိုက်ခြင်းအားဖြင့် တစ်ပိုင်းချင်းစီကို သီးသန့်ရေးသားနိုင်ပြီး၊ အမှားရှာဖွေရ လွယ်ကူစေပါတယ် (Debugging is easier) ။

Chapter 3: Software Development Life Cycle (SDLC)

(ဆော့ဖ်ဝဲ တည်ဆောက်ရေး လုပ်ငန်း)

3.1 SDLC ဆိုတာ ဘာလဲ?

ဆော့ဖ်ဝဲတစ်ခု ဖြစ်ပေါ်လာပုံကို လူတစ်ယောက်ရဲ့ ဘဝသံသရာ (မွေးဖွား၊ ကြီးပြင်း) လိုမျိုး အဆင့်ဆင့် သတ်မှတ်ထားတာကို **Software Development Life Cycle (SDLC)** လို့ခေါ်ပါတယ် ။

ဘယ် Model ကိုပဲသုံးသုံး SDLC မှာ မဖြစ်မနေ ပါဝင်တဲ့ အဓိက အဆင့် (၅) ဆင့် ရှိပါတယ်:

1. **Planning (စီမံကိန်း ရေးဆွဲခြင်း):** ဘာကြောင့် ဒီဆော့ဖ်ဝဲကို ရေးမှာလဲ? ဘယ်သူတွေ သုံးမှာလဲ? ငွေဘယ်လောက် ကုန်မလဲ? ဆိုတာကို တွက်ချက်တဲ့ အဆင့်ပါ ။
2. **Analysis (လိုအပ်ချက်ကို ဆန်းစစ်ခြင်း):** အသုံးပြုသူတွေ (Users) တကယ်လိုချင်တာ ဘာလဲဆိုတာကို မေးမြန်းစစ်ဆေးပြီး စာရင်းပြုစုခြင်း ဖြစ်ပါတယ် ။
3. **Design (ပုံစံထုတ်ခြင်း):** ကုဒ်မရေးခင် "ဒီဇိုင်း" အရင်ဆွဲရပါတယ်။
   Logical Design: ဆော့ဖ်ဝဲရဲ့ လုပ်ဆောင်ချက်တွေ (Screen ပုံစံ၊ Database ပုံစံ) ကို စာရွက်ပေါ်မှာ ပုံဖော်တာပါ ။
4. **Implementation (လက်တွေ့ တည်ဆောက်ခြင်း):** ဒီအဆင့်ကျမှ Programmer တွေက ကုဒ် (Coding) စရေးတာ ဖြစ်ပါတယ်။

5. **Maintenance (ထိန်းသိမ်း ပြုပြင်ခြင်း):** ဆော့ဖ်ဝဲထွက်ပြီးသွားရင် ပြီးမသွားပါဘူး။ Error တက်ရင် ပြင်ပေးရတယ်၊ Feature အသစ်တွေ ထပ်ထည့်ပေးရပါတယ် ။

3.2 Waterfall Model

ဒါဟာ ရှေးအကျဆုံးနဲ့ အခြေခံအကျဆုံး SDLC ပုံစံ ဖြစ်ပါတယ် ။

(a) သဘောတရား

ရေတံခွန်က ရေတွေဟာ အပေါ်ကနေ အောက်ကိုပဲ စီးဆင်းသလိုပါပဲ။ အဆင့်တစ်ခု ပြီးမှ နောက်တစ်ဆင့်ကို ကူးပြောင်းလို့ ရပါတယ်။ (ဥပမာ - Design အဆင့် မပြီးမချင်း Coding စလို့မရပါဘူး) ။

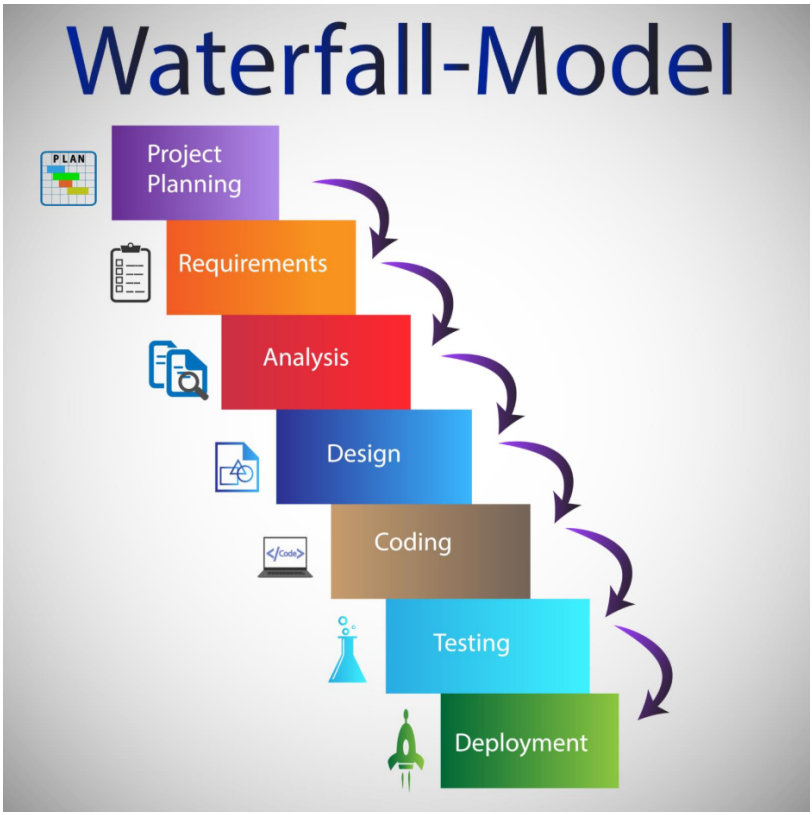


image - Shutterstock

(b) အားသာချက်များ

- အစကတည်းက ဘာလုပ်ရမယ်ဆိုတာ တိကျပြီးသားဖြစ်လို့ စီမံခန့်ခွဲရ လွယ်ကူပါတယ် ။
- စာချုပ်ချုပ်ပြီး လုပ်ရတဲ့ အစိုးရ Project တွေ၊ ဆောက်လုပ်ရေး လုပ်ငန်းခွင်သုံး စနစ်တွေမှာ သုံးလေ့ရှိပါတယ်။

(c) အားနည်းချက်များ

- **ပြောင်းလဲရ ခက်ခဲခြင်း:** Coding ရေးနေတုန်းမှ User က "ဒီနေရာလေး ပြင်ချင်တယ်" လို့ ပြောရင် ပြင်ဖို့ အရမ်းခက်ပါတယ်။ အစကနေ ပြန်သွားရသလိုဖြစ်တတ်ပါတယ်။
- **ရလဒ်မြင်ရ ကြာခြင်း:** စီမံကိန်း ပြီးဆုံးခါနီးမှ ဆော့ဖ်ဝဲကို မြင်ရတဲ့အတွက်၊ လမ်းမှားရောက်နေရင် ပြန်ပြင်ဖို့ နောက်ကျသွားတတ်ပါတယ်။

3.3 Agile Model (ခေတ်သစ် ပေါ့ပါးသွက်လက်သော ပုံစံ)

ယနေ့ခေတ် Facebook, Google တို့လို နည်းပညာ ကုမ္ပဏီတွေက Waterfall ကို မသုံးကြတော့ပါဘူး။ အပြောင်းအလဲကို လက်ခံနိုင်တဲ့ **Agile** နည်းလမ်းကို သုံးကြပါတယ်။

(a) သဘောတရား

ဆော့ဖ်ဝဲ အကြီးကြီးတစ်ခုလုံးကို တစ်ခါတည်း အပြီးမရေးပါဘူး။ အပိုင်းငယ်လေးတွေ (Sprints) ခွဲပြီး တစ်ပိုင်းချင်းစီ ရေးပါတယ်။

- **Sprint:** ပုံမှန်အားဖြင့် ၂ ပတ် ကြာပါတယ်။
- ၂ ပတ်နေရင် ရလာတဲ့ ရလဒ်အသေးစားလေးကို User ကို ပြလိုက်ပါတယ်။ User သဘောကျမှ နောက်တစ်ပိုင်း ဆက်ရေးပါတယ်။

(b) Scrum (Agile ၏ နည်းလမ်းတစ်ခု)

Agile သဘောတရားကို လက်တွေ့လုပ်ဆောင်တဲ့ နည်းလမ်းကို **Scrum** လို့ခေါ်ပါတယ်။

- **Stand-up Meeting:** မနက်တိုင်း ၁၅ မိနစ် မတ်တပ်ရပ်ပြီး "မနေ့က ဘာပြီးလဲ၊ ဒီနေ့ ဘာလုပ်မလဲ" ဆိုတာ အဖွဲ့သားတွေ တိုင်ပင်ကြပါတယ်။

3.4 Prototyping (နမူနာပြ ပုံစံ)

User က သူဘာလိုချင်မှန်း သူကိုယ်တိုင် သေချာမသိတဲ့အခါမျိုးမှာ သုံးပါတယ် ။

- **နည်းလမ်း:** ကုဒ်တွေ အများကြီး မရေးဘဲ၊ အပေါ်ယံ အစမ်းသုံးလို့ရတဲ့ **Prototype (နမူနာ)** ကို အမြန်ဖန်တီးလိုက်ပါတယ်။
- **ရည်ရွယ်ချက်:** User ကို ပြသပြီး Feedback ယူဖို့ သက်သက် ဖြစ်ပါတယ်။ အတည်ယူသုံးဖို့ မဟုတ်ပါဘူး ။

Chapter 4: System Modeling

(စနစ်ပုံဖော်ခြင်းနှင့် ပုံဆွဲနည်းလမ်းများ)

4.1 System Modeling ဆိုတာ ဘာလဲ?

System Modeling ဆိုတာ စနစ်တစ်ခုလုံးရဲ့ အလုပ်လုပ်ပုံကို စာတွေချည်း ရေးနေမယ့်အစား၊ ရုပ်ပုံတွေ၊ ဇယားတွေနဲ့ ရှင်းလင်းစွာ ပြသခြင်း ဖြစ်ပါတယ်။ အဓိက ရည်ရွယ်ချက်ကတော့ -

- **Communication:** ပရိုဂရမ်မာနဲ့ အသုံးပြုသူ (User) ကြား နားလည်မှုလွဲတာတွေ မဖြစ်ရအောင် ပုံနဲ့ ညှိနှိုင်းဖို့ပါ။

- **Blueprint:** ကုဒ်ရေးတဲ့အခါ လမ်းမပျောက်ရအောင် လမ်းညွှန်မြေပုံ အဖြစ် သုံးဖို့ပါ။

4.2 Data Flow Diagram - DFD (အချက်အလက် စီးဆင်း ပုံပြ ဇယား)

DFD ဆိုတာ စနစ်ထဲမှာ အချက်အလက် (Data) တွေ ဘယ်ကဝင်လာပြီး၊ ဘယ်ကိုရောက်သွားလဲ ဆိုတာကို ပြတဲ့ ပုံဖြစ်ပါတယ်။ Flowchart နဲ့ မတူတာက သူက "အဆင့်" (Step) ကို မပြဘဲ "ဒေတာသွားပုံ" ကိုပဲ အဓိကထား ပြပါတယ်။

DFD မှာ အဓိက သင်္ကေတ (၄) ခု ပါဝင်ပါတယ် -

(a) External Entity

- **သင်္ကေတ:** လေးထောင့်ကွက် (Square)။
- **အဓိပ္ပာယ်:** စနစ်ကို အသုံးပြုသူတွေ သို့မဟုတ် စနစ်နဲ့ လှမ်းချိတ်ထားတဲ့ တခြားစနစ်တွေ ဖြစ်ပါတယ်။
- **ဥပမာ:** Customer (ဖောက်သည်)၊ Bank System (ဘဏ်စနစ်)။

(b) Process (လုပ်ငန်းစဉ်)

- **သင်္ကေတ:** စက်ဝိုင်း သို့မဟုတ် ထောင့်ဝိုင်းလေးထောင့် (Circle/ Rounded Rectangle)။
- **အဓိပ္ပာယ်:** Data ကို ပုံစံပြောင်းပေးလိုက်တဲ့ အလုပ် (Action)။
- **ဥပမာ:** "အတိုးနှုန်း တွက်ချက်ခြင်း"၊ "ဘေလ် ထုတ်ပေးခြင်း”။

(c) Data Store (အချက်အလက်သိမ်းဆည်း)

- **သင်္ကေတ:** ထိပ်ပွင့်နေတဲ့ လေးထောင့်ကွက် (Open-ended Rectangle)။
- **အဓိပ္ပာယ်:** Data တွေကို သိမ်းဆည်းထားတဲ့ နေရာ (Database သို့မဟုတ် ဖိုင်)။
- **ဥပမာ:** Customer စာရင်း၊ ပစ္စည်းစာရင်း။

(d) Data Flow (ဒေတာ လမ်းကြောင်း)

- **သင်္ကေတ:** မြှား (Arrow)။
- **အဓိပ္ပာယ်:** Data တွေ ဘယ်ကနေ ဘယ်ကို စီးဆင်းသွားလဲ ဆိုတာ ပြပါတယ်။
- **ဥပမာ:** အော်ဒါအချက်အလက် (Order Details)။

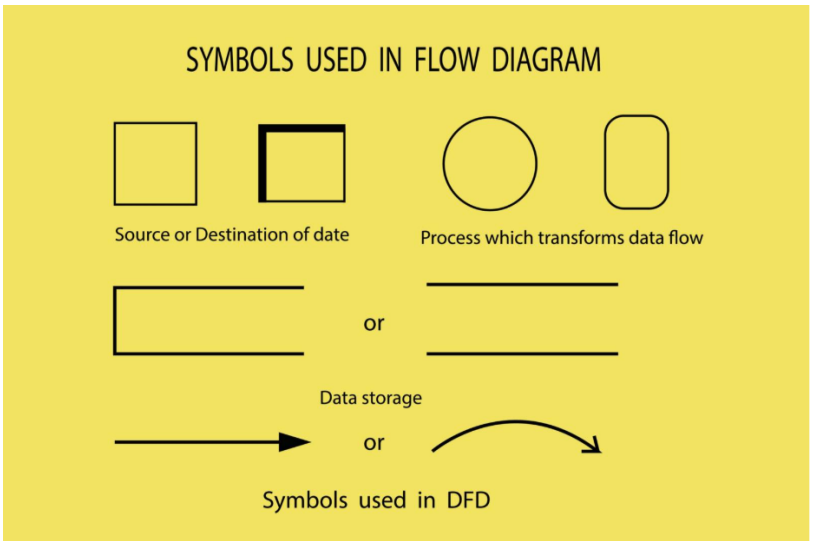


image - Shutterstock

4.3 Decision Tables (ဆုံးဖြတ်ချက် ဇယားများ)

Logic တွေ အရမ်းရှုပ်ထွေးနေတဲ့အခါ (ဥပမာ - အခြေအနေ ၅ မျိုးလောက်ကို ချိန်ညှိပြီးမှ ဆုံးဖြတ်ရမယ်ဆိုရင်) Flowchart ဆွဲရင် ရှုပ်သွားတတ်ပါတယ်။ ဒီလိုအခါမျိုးမှာ **Decision Table** ကို သုံးပါတယ်။

ဇယားဖွဲ့စည်းပုံ:

ဇယားကို (၄) ပိုင်း ခွဲထားပါတယ်:

1. **Condition Stubs:** ဖြစ်နိုင်ခြေရှိတဲ့ အခြေအနေများ (ဥပမာ - မိုးရွာလား? ထီးပါလား?)။
2. **Action Stubs:** လုပ်ဆောင်ရမယ့် အလုပ်များ (ဥပမာ - အပြင်ထွက်မယ်၊ အိမ်မှာနေမယ်)။
3. **Rules:** အခြေအနေနဲ့ အလုပ်ကို တွဲစပ်ပေးတဲ့ စည်းမျဉ်းများ (Y/N အမှန်အမှားများ)။

ဥပမာ - တက္ကသိုလ် ဝင်ခွင့်စစ်ဆေးခြင်း :

Conditions (အခြေအနေ)	Rule 1	Rule 2	Rule 3
စာမေးပွဲ အမှတ် ၅၀၀ ကျော်လား?	Yes	No	Yes
အကျင့်စာရိတ္တ ကောင်းလား?	Yes	Yes	No
Actions (ရလဒ်)			
ကျောင်းဝင်ခွင့်ပြုသည်	X		
ဝင်ခွင့် ငြင်းပယ်သည်		X	X

4.4 Unified Modeling Language - UML (ခေတ်သစ် စံပြုပုံစံ)

ယနေ့ခေတ် Modern Software Development မှာ DFD ထက် **UML** ကို ပိုသုံးလာကြပါပြီ။ UML ဆိုတာ ကမ္ဘာသုံး စံသတ်မှတ်ထားတဲ့ ပုံဆွဲနည်းစနစ် ဖြစ်ပါတယ်။

(a) Use Case Diagram (အသုံးပြုမှု ပုံဖော်ခြင်း)

ရည်ရွယ်ချက်: "ဘယ်သူက ဘာလုပ်မှာလဲ" (Who does What?) ဆိုတာကို ပြပါတယ်။

ပါဝင်သူများ:

- **Actor:** လူရုပ်ပုံလေးနဲ့ ပြပါတယ်။ (ဥပမာ - User, Admin)။
- **Use Case:** ဘဲဥပုံလေးနဲ့ ပြပါတယ်။ (ဥပမာ - Login ဝင်ခြင်း၊ ပိုက်ဆံလွှဲခြင်း)။

(b) Class Diagram (ဖွဲ့စည်းပုံ ပုံဖော်ခြင်း)

- **ရည်ရွယ်ချက်:** ကုဒ်ရေးမယ့် ပရိဂရမ်မာတွေအတွက် စနစ်ရဲ့ အတွင်းပိုင်း တည်ဆောက်ပုံကို ပြပါတယ်။
- **ပါဝင်သူများ:** Class တွေ၊ သူတို့ရဲ့ ဂုဏ်သတ္တိ (Attributes) တွေ နဲ့ လုပ်ဆောင်ချက် (Methods) တွေ ပါဝင်ပါတယ်။ (ဒါက Object-Oriented Programming နဲ့ တိုက်ရိုက် သက်ဆိုင်ပါတယ်)။

Chapter 5: Data Modeling & Databases

(အချက်အလက် ပုံဖော်ခြင်းနှင့် ဒေတာဘေ့စ် နည်းပညာ)

5.1 Data Modeling ဆိုတာ ဘာလဲ?

အိမ်မဆောက်ခင် အိမ်ပုံစံ (Architectural Plan) ရှိဖို့ လိုသလိုပဲ၊ Database တစ်ခု မတည်ဆောက်ခင်မှာလည်း **Data Model** ရှိဖို့ လိုအပ်ပါတယ်။
Data Modeling ဆိုတာ စနစ်တစ်ခုမှာ ဘယ်လို အချက်အလက်တွေ (Entities) ရှိမယ်၊ သူတို့က ဘယ်လို ဆက်စပ်နေလဲ (Relationships) ဆိုတာကို ပုံဖော်ခြင်း ဖြစ်ပါတယ် ။

- **Logical Data Model:** Data တွေကို ဘယ်လိုဖွဲ့စည်းထားမလဲဆိုတာကို အာရုံစိုက်ပါတယ် (နည်းပညာပိုင်း မပါဝင်ပါ) ။
- **Physical Data Model:** တကယ်တမ်း ကွန်ပျူတာထဲမှာ ဘယ်လိုသိမ်းမလဲဆိုတာကို ပြသပါတယ် ။

5.2 Entity-Relationship Diagram (ERD)

Database ဒီဇိုင်းဆွဲရာမှာ လူသုံးအများဆုံး ပုံဆွဲနည်းကတော့ **ER Diagram** ဖြစ်ပါတယ်။ သူ့မှာ အဓိက အစိတ်အပိုင်း (၃) ခု ပါဝင်ပါတယ်:

(a) Entity (အရာဝတ္ထု)

- သိမ်းဆည်းချင်တဲ့ အကြောင်းအရာ တစ်ခုခု ဖြစ်ပါတယ်။
- **သင်္ကေတ:** လေးထောင့်ကွက် (Rectangle)။
- **ဥပမာ:** Customer (ဝယ်သူ), Product (ကုန်ပစ္စည်း)။

(b) Attribute (ဂုဏ်သတ္တိ)

- Entity တစ်ခုချင်းစီရဲ့ အသေးစိတ် အချက်အလက်တွေ ဖြစ်ပါတယ်။
- **သင်္ကေတ:** ဘဲဥပုံ (Oval)။
- **ဥပမာ:** Customer Name, Phone Number, Address။

(c) Relationship (ဆက်နွယ်မှု)

- Entity တစ်ခုနဲ့ တစ်ခု ဘယ်လို ပတ်သက်နေလဲ ဆိုတာကို ပြပါတယ်။
- **သင်္ကေတ:** စိန်ပွင့်ပုံ (Diamond)။
- **ဥပမာ:** Customer တစ်ယောက်က Product ကို Buys (ဝယ်ယူသည်)။

5.3 Normalization (အချက်အလက်များကို သန့်စင်ခြင်း)

Database တစ်ခု တည်ဆောက်တဲ့အခါ Data တွေ ထပ်နေတာ (Redundancy)၊ ရှုပ်ထွေးနေတာတွေ မဖြစ်အောင် **Normalization** လုပ်ရပါတယ် ။ အဆင့် (၃) ဆင့်နဲ့ ရှင်းပြပါမယ် -

(a) First Normal Form (1NF) – အထပ်ထပ် မဖြစ်စေရ

ဇယားကွက်တစ်ခုထဲမှာ တန်ဖိုးတွေ အများကြီး မထည့်ရပါဘူး။

- **မကောင်းတဲ့ ဥပမာ:** "Mg Mg" က "Apple, Orange" ဝယ်တယ်လို့ တစ်ကြောင်းတည်း ရေးထားတာ။

CustomerName	Products
Mg Mg	Apple, Orange
Aye Aye	Banana

- **1NF ပြောင်းခြင်း:** "Mg Mg - Apple" တစ်ကြောင်း၊ "Mg Mg - Orange" တစ်ကြောင်း ခွဲရေးရပါမယ် ။

CustomerName	Product
Mg Mg	Apple
Mg Mg	Orange
Aye Aye	Banana

(b) Second Normal Form (2NF) – အမှီအခို ကင်းစေရ

Primary Key ကို အပြည့်အဝ မှီခိုနေတဲ့ Data တွေပဲ ရှိရပါမယ် ။

- **ဥပမာ:** Order ID နဲ့ Product ID တွဲမှ ဈေးနှုန်းသိရမယ့် ဇယား မှာ Customer Address သွားထည့်ထားရင် မဆိုင်ပါဘူး။ Customer Address ကို Customer ဇယား သက်သက် ခွဲထုတ်လိုက်ပါ။

CustomerID	CustomerName
1	Mg Mg
2	Aye Aye

CustomerID	Product
1	Apple
1	Orange
2	Banana

(c) Third Normal Form (3NF) – သွယ်ဝိုက်မှု မရှိစေရ

Key မဟုတ်တဲ့ တခြား Data အချင်းချင်း မှီခိုနေတာမျိုး မရှိရပါဘူး ။

- **ဥပမာ:** Order ဇယားထဲမှာ Customer ID ရှိရင် ရပါပြီ။ Customer Name ပါ ထပ်ထည့်စရာ မလိုပါဘူး (ID ကနေတဆင့် Name ကို လှမ်းကြည့်လို့ ရပြီးသားမို့လို့ပါ)။

CustomerID	CustomerName
1	Mg Mg
2	Aye Aye

ProductID	ProductName	Price
1	Apple	1000
2	Orange	800
3	Banana	500

CustomerID	ProductID
1	1
1	2
2	3

5.4 File Organization & Database (ဖိုင်မှ ဒေတာဘေ့စ်သို့)

(a) Traditional File System (ရှေးရိုး ဖိုင်စနစ်)

အရင်ခေတ်က Data တွေကို Text File တွေအနေနဲ့ သိမ်းခဲ့ကြပါတယ်။

- **Sequential Access:** ရှေ့ကနေ နောက်ကို တစ်ခုပြီးတစ်ခု လိုက်ရှာရတဲ့ စနစ် (ကက်ဆက်ခွေ လိုမျိုးပါ) ။
- **Direct Access:** လိုချင်တဲ့နေရာကို တန်းသွားလို့ရတဲ့ စနစ် (ဓာတ်ပြား သို့မဟုတ် CD လိုမျိုးပါ) ။

(b) Modern DBMS (ခေတ်သစ် ဒေတာဘေ့စ် စီမံခန့်ခွဲမှု)

ယနေ့ခေတ်မှာတော့ File တွေကို ကိုယ်တိုင်စီမံမနေတော့ဘဲ **DBMS (Database Management System)** ဆော့ဖ်ဝဲတွေကို သုံးလာကြပါတယ်။

- **Relational Database (SQL):** ဇယား (Tables) တွေနဲ့ သိမ်းဆည်းတဲ့ စနစ်။ (ဥပမာ - MySQL, PostgreSQL, Microsoft SQL Server)။
- **NoSQL Database:** ဇယားမဆောက်ဘဲ လွတ်လပ်စွာ သိမ်းဆည်းတဲ့ စနစ် (ဥပမာ - MongoDB, Firebase)။ Mobile App တွေနဲ့ Real-time Chat တွေမှာ အသုံးများပါတယ်။

SQL Corner: Database ကို ခိုင်းစေခြင်း

Data တွေကို စီမံခန့်ခွဲဖို့အတွက် **SQL (Structured Query Language)** ကို သုံးရပါတယ်။ အင်္ဂလိပ်စာနဲ့ ဆင်တူပါတယ်။

Database လုပ်ဆောင်ချက် အဓိက (၄) မျိုးရှိပြီး ယေဘုယျအားဖြင့် **CRUD** လို့ ခေါ်ကြပါတယ်။

1. Create (အသစ်ထည့်ခြင်း – INSERT)

ကျောင်းသားအသစ် တစ်ယောက် ထည့်ချင်ရင် -

```
INSERT INTO Students (Name, Age)
VALUES ('Mg Ba', 19);
```

2. Read (ဖတ်ရှုခြင်း – SELECT)

အသက် ၁၈ နှစ်အထက် ကျောင်းသားတွေကို ရှာပေးပါလို့ ခိုင်းချင်ရင် -

```
SELECT * FROM Students
WHERE Age > 18;
```

- SELECT * : အကုန်ယူမယ်။
- FROM Students: Students ဆိုတဲ့ ဇယားထဲကနေ။
- WHERE Age > 18: အသက် ၁၈ ကြီးမှ။

3. Update (ပြင်ဆင်ခြင်း – UPDATE)

Mg Ba ရဲ့ အသက်ကို ၂၀ လို့ ပြင်ချင်ရင် -

```
UPDATE Students
SET Age = 20
WHERE Name = 'Mg Ba';
```

4. Delete (ဖျက်ခြင်း – DELETE)

Mg Ba ကို စာရင်းထဲက ဖျက်ချင်ရင် -

```
DELETE FROM Students
WHERE Name = 'Mg Ba’;
```

မှတ်သားရန် (Key Concept):
Database ဇယားတွေမှာ လူနာမည်တွေ တူနေရင် မှားယွင်းမှုမရှိအောင် တစ်ယောက်နဲ့တစ်ယောက် မတူနိုင်တဲ့ နံပါတ်တစ်ခု သတ်မှတ်ပေးရပါတယ်။ အဲ့ဒါကို **Primary Key (ဥပမာ – ID Number, Roll Number)** လို့ခေါ်ပါတယ်။

*"ဤစာအုပ်များသည် တက္ကသိုလ်များတွင် သင်ကြားပို့ချနေသော Computer Science သင်ရိုးညွှန်းတမ်း (Curriculum) များကို အခြေခံထားပါသည်။ သို့သော် ယနေ့ခေတ် နည်းပညာများနှင့် ကိုက်ညီစေရန် အတွက် Python ဘာသာစကား၊ ခေတ်မီ Hardware နည်းပညာများ နှင့် လက်တွေ့လုပ်ငန်းခွင်သုံး ဥပမာများကို ဖြည့်စွက် ပြုပြင်ရေးသားထားခြင်း ဖြစ်ပါသည်။

မူရင်းသင်ရိုးကို ပြုစုခဲ့ကြသော ဆရာ/ဆရာမ များအားလုံးကို လေးစားစွာဖြင့် Credit ပေးပါသည်။"*

Yan Naing Lin
Software Engineer