

252SE311: ວິສະວະກຳຊອບແວຣ໌



ການກຳໜົດຄວາມຕ້ອງການ ແລະ ອອກແບບ

ບົດທີ 9

ການອອກແບບຊອບແວຣ໌

ເນື້ອໃນຫຍໍ້

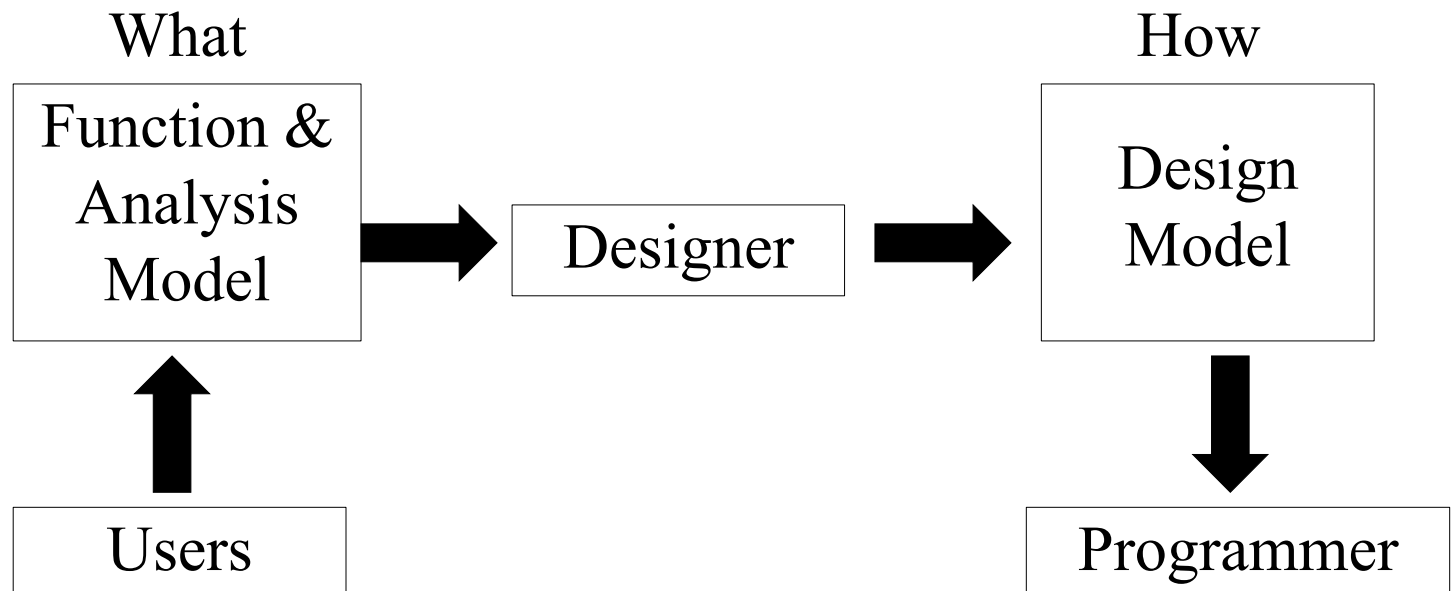


- ◆ ຄວາມໝາຍຂອງການອອກແບບຊອບແວຣ໌
- ◆ ຂະບວນການອອກແບບຊອບແວຣ໌
- ◆ ສະຖາປັດຕະຍະກຳ ແລະ ໂຄງສ້າງສະຖາປັດຕະຍະກຳຊອບແວຣ໌
- ◆ ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການອອກແບບຊອບແວຣ໌
- ◆ ແນວຄິດໃນການອອກແບບຊອບແວຣ໌
- ◆ ວິທີການ ແລະ ຫຼັກການຂອງການອອກແບບຊອບແວຣ໌
- ◆ ແບບຈຳລອງທີ່ໃຊ້ໃນການອອກແບບ

ຄວາມໝາຍຂອງການອອກແບບຊອບແວຣ໌

↪ ການອອກແບບລະບົບ

- ເປັນການເອົາຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ມາປ່ຽນເປັນແມ່ແບບ
- ສິ່ງທີ່ໄດ້ຈາກການອອກແບບແມ່ນ ເອກະສານຂໍ້ກຳໜົດກ່ຽວກັບການອອກແບບ (Design Specification Document) ທີ່ປະກອບດ້ວຍແບບຈຳລອງຂອງສ່ວນປະກອບທີ່ຈະລວມເຂົ້າເປັນລະບົບ



ຄວາມໝາຍຂອງການອອກແບບຊອບແວຣ໌

➤ ການອອກແບບຊອບແວຣ໌

- ແມ່ນຂະບວນການກຳໜົດສະຖາປັດຕະຍະກຳ, ສ່ວນປະກອບ, ສ່ວນປະສານງານ ແລະ ລັກສະນະອື່ນໆຂອງລະບົບ ຊຶ່ງເປັນການເອົາຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ມາກຳໜົດລາຍລະອຽດໂຄງສ້າງພາຍໃນຂອງຊອບແວຣ໌
- ສິ່ງທີ່ໄດ້ຈາກການອອກແບບຄື ແບບຈຳລອງຂອງການອອກແບບ (Design Model)

ຂະບວນການອອກແບບຊອບແວ

- ຂະບວນການອອກແບບຊອບແວຈະເປັນການເຮັດວຽກແບບຊ້າໆ
- ຂະບວນການອອກແບບຊອບແວແບ່ງອອກເປັນ 2 ລະດັບ
 - ການອອກແບບສະຖາປັດຕະຍະກຳ (Architectural Design)
 - ເປັນການກຳນົດລັກສະນະໂຄງສ້າງຂອງລະບົບ ຫຼື ຊອບແວ ຊຶ່ງສະແດງໃຫ້ເຫັນສ່ວນປະກອບຕ່າງໆຂອງຊອບແວໃນການເບິ່ງລະດັບເທິງ (ເປັນການສະແດງໃຫ້ເຫັນສ່ວນປະກອບຕ່າງໆຂອງຊອບແວ)
 - ການອອກແບບໃນລາຍລະອຽດ (Detailed Design)
 - ເປັນການອະທິບາຍລາຍລະອຽດຂອງແຕ່ລະສ່ວນປະກອບຂອງຊອບແວເພື່ອສະດວກໃຫ້ແກ່ການຂຽນໂປຣແກຣມ

ສະຖາປັດຕະຍະກຳ ແລະ ໂຄງສ້າງ

ສະຖາປັດຕະຍະກຳຊອບແວ

- ສະຖາປັດຕະຍະກຳຊອບແວໝາຍເຖິງການອະທິບາຍການພົວພັນລະຫວ່າງລະບົບຍ່ອຍ ແລະ ສ່ວນປະກອບຂອງມັນເພື່ອກຳນົດໂຄງສ້າງທີ່ລະບົບພາຍໃນຊອບແວ
- ໂຄງສ້າງສະຖາປັດຕະຍະກຳແມ່ນການກຳນົດລາຍລະອຽດໃນແຕ່ລະດ້ານຂອງຊອບແວແລ້ວເອົາມາປະກອບເຂົ້າກັນເປັນຊອບແວ
- ໂຄງສ້າງສະຖາປັດຕະຍະກຳໄດ້ແບ່ງຮູບແບບຂອງສະຖາປັດຕະຍະກຳ (Architecture Style) ອອກເປັນຫລາຍກຸ່ມ, ແຕ່ລະກຸ່ມມີລັກສະນະໂຄງສ້າງ ແລະ ຈຸດປະສົງທີ່ແຕກຕ່າງກັນ
- ຮູບແບບຂອງສະຖາປັດຕະຍະກຳເປັນຂໍ້ບັງຄັບທີ່ກົດເກນທາງດ້ານສະຖາປັດຕະຍະກຳທີ່ສ້າງຂຶ້ນມາເພື່ອຈຳແນກກຸ່ມທີ່ໝວດໝູ່ຂອງສະຖາປັດຕະຍະກຳຊອບແວ

ສະຖາປັດຕະຍະກຳ ແລະ ໂຄງສ້າງ ສະຖາປັດຕະຍະກຳຊອບແວ

- ຮູບແບບສະຖາປັດຕະຍະກຳແບ່ງອອກເປັນ 5 ຮູບແບບ
- ສະຖາປັດຕະຍະກຳແບບໂຄງສ້າງທົ່ວໄປ (Layer, Pipe and Filter, Blackboard)
 - ສະຖາປັດຕະຍະກຳລະບົບແບບກະຈາຍ (Client/Server, Three Tiers, Broker)
 - ສະຖາປັດຕະຍະກຳລະບົບແບບ Interactive (Model-View-Controller, Presentation-Abstraction-Control)
 - ສະຖາປັດຕະຍະກຳລະບົບທີ່ສາມາດດັດແປງໄດ້ (Micro-Kernel, Reflection)
 - ສະຖາປັດຕະຍະກຳລະບົບແບບອື່ນໆ (Batch, Interpreter, Process Control, Rule Based)

ສະຖາປັດຕະຍະກຳ ແລະ ໂຄງສ້າງ ສະຖາປັດຕະຍະກຳຊອບແວ

↳ ແບບແຜນການອອກແບບ (Design Pattern)

- ແບບແຜນ (Pattern) ແມ່ນວິທີແກ້ບັນຫາທີ່ມີລັກສະນະເປັນກາງ ເພື່ອໃຊ້ກັບບັນຫາທີ່ໄປ
- Design Pattern ໃນລະດັບຈຸນລະພາກແບ່ງອອກເປັນ 3 ກຸ່ມ
 - Creational Pattern (Builder, Factory, Prototype, Singleton)
 - Structural Pattern (Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy)
 - Behavioral Pattern (Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template, Visitor)

ສະຖາປັດຕະຍະກຳ ແລະ ໂຄງສ້າງ ສະຖາປັດຕະຍະກຳຊອບແວ

➤ ກຸ່ມຂອງຊອບແວ ແລະ Framework

- ວິທີທີ່ຊ່ວຍສະໜັບສະໜູນການອອກແບບຊອບແວ ແລະ ບັນດາສ່ວນປະກອບຂອງຊອບແວໃຫ້ສາມາດເອົາກັບມາໃຊ້ໃໝ່ໄດ້ແມ່ນການກຳໜົດເປັນກຸ່ມຂອງຊອບແວ (Families of Software) ຊຶ່ງເອີ້ນວ່າ Software Product Line
- ຊອບແວທີ່ມີລັກສະນະບາງຢ່າງຄືກັນຈະຈັດໄວ້ໃນກຸ່ມດຽວກັນ ແລະ ຊອບແວທີ່ຢູ່ໃນກຸ່ມດຽວກັນຈະສາມາດນຳກັບມາໃຊ້ໃໝ່ໄດ້ໂດຍດັດແປງພຽງເລັກໜ້ອຍເທົ່ານັ້ນ
- ຊຶ່ງຊອດຄ່ອງກັບວິທີການຂຽນໂປຣແກຣມໂດຍໃຊ້ແນວຄິດທີ່ເອີ້ນວ່າ Framework

ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການ ອອກແບບຊອບແວ

↳ ຄຸນລັກສະນະຂອງຄຸນນະພາບ (Quality Attribute)

- ການເຮັດວຽກຂອງໂປຣແກຣມ (Functionality)
 - ເບິ່ງຈາກ ຈຳນວນ Feature ແລະ ຄວາມສາມາດຂອງໂປຣແກຣມ
 - ເບິ່ງຈາກໜ້າທີ່ທີ່ໄປຂອງໂປຣແກຣມ ແລະ ຄວາມປອດໄພ
- ຄວາມສາມາດໃນການໃຊ້ງານ (Usability)
 - ເບິ່ງຈາກການ feedback ຈາກການໃຊ້ງານຂອງຜູ້ໃຊ້ວ່າມັນໃຊ້ງ່າຍບໍ່ ແລະ ຮຽນຮູ້ງ່າຍບໍ່
- ຄວາມໜ້າເຊື່ອຖື (Reliability)
 - ເບິ່ງຈາກການນັບຄວາມຖີ່ ແລະ ຄວາມຮຸນແຮງຂອງຄວາມຜິດພາດທີ່ເກີດຂຶ້ນ, ຄວາມຖືກຕ້ອງຂອງຜົນຮັບທີ່ໄດ້, ເວລາສະເລ່ຍຂອງຄວາມບໍ່ສໍາເລັດ, ຄວາມສາມາດໃນການກູ້ຄືນລະບົບ ແລະ ຄວາມສາມາດໃນການຄາດການໄດ້ຂອງໂປຣແກຣມ

ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການ ອອກແບບຊອບແວຣ໌

↳ ຄຸນລັກສະນະຂອງຄຸນນະພາບ (Quality Attribute)

- ປະສິດທິພາບ (Performance)
 - ເບິ່ງຈາກຄວາມໄວຂອງການປະມວນຜົນ, ໄລຍະເວລາຕອບສະໜອງ, ຊັບພະຍາກອນທີ່ໃຊ້, ປະລິມານການປະຕິບັດໄດ້ໃນຊ່ວງເວລາໃດໜຶ່ງ ແລະ ປະສິດທິຜົນໃນການເຮັດວຽກ
- ຄວາມສາມາດໃນການສະໜັບສະໜູນການໃຊ້ງານ (Supportability) ແລະ ຄວາມສາມາດໃນການບໍາລຸງຮັກສາ
 - ເບິ່ງຈາກຄວາມສາມາດໃນການເຮັດວຽກເພີ່ມເຕີມ, ຄວາມສາມາດໃນການດັດແປງການເຮັດວຽກ ແລະ ການບໍລິການ, ຄວາມສາມາດໃນການທົດສອບ, ການເຮັດວຽກຂ້າມລະບົບ ແລະ ການຈັດສັນສະພາບແວດລ້ອມຂອງລະບົບນຳອີກ

ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການ ອອກແບບຊອບແວຣ໌

- ວິເຄາະ ແລະ ປະເມີນຄຸນນະພາບ (Quality Analysis and Evaluation)
- ການວິເຄາະ ແລະ ປະເມີນຄຸນນະພາບເປັນກິດຈະກຳທີ່ຊ່ວຍໃຫ້ໝັ້ນໃຈວ່າ ຊອບແວຣ໌ທີ່ໄດ້ອອກແບບໄວ້ມີຄຸນນະພາບ
 - ການວິເຄາະແລະປະເມີນໄດ້ແບ່ງຕາມກິດຈະກຳດັ່ງນີ້
 - ທົບທວນຄືນການອອກແບບຊອບແວຣ໌ (Software Design Review)
 - ວິເຄາະການອອກແບບ (Static Analysis)
 - ການຈຳລອງສະຖານະການ ແລະ ການສ້າງແບບຈຳລອງ (Simulation and Prototyping)

ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການ ອອກແບບຊອບແວຣ໌

➤ ການວັດແທກ (Measure)

- ສໍາຫລັບການວັດແທກຄຸນນະພາບຂອງການອອກແບບຊອບແວຣ໌ ປະເພດຂອງການວັດແທກຂຶ້ນຢູ່ກັບວິທີການໃນການອອກແບບທີ່ເລືອກໃຊ້ ໂດຍແບ່ງອອກເປັນ 2 ປະເພດ:
 - ການວັດແທກການອອກແບບແບບ Function
 - ສາມາດວັດແທກຄຸນນະພາບໄດ້ຈາກໄດ້ຈາກຄຸນລັກສະນະຂອງ Module ເຊັ່ນ: Coupling ແລະ Cohesion
 - ການວັດແທກການອອກແບບທາງວັດຖຸ
 - ການວັດແທກຄຸນນະພາບສາມາດເຮັດໄດ້ດ້ວຍລັກສະນະພາຍໃນ Class ເຊັ່ນ: ການວັດແທກຄວາມສໍາພັນລະຫວ່າງ Class, ການນັບຈໍານວນການໂຕ້ຕອບກັນລະຫວ່າງ Method ຂອງ Class

ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການ ອອກແບບຊອບແວ

➤ ຫຼັກການອອກແບບຊອບແວ

- ແນວຄິດການອອກແບບເພື່ອນຳໄປສູ່ການອອກແບບທີ່ດີ
 1. ການອອກແບບຄວນສະແດງໃຫ້ເຫັນເຖິງຮູບແບບສະຖາປັດຕະຍະກຳທີ່ເລືອກໃຊ້ຢ່າງຊັດເຈນ ແລະ ມີແບບແຜນ
 2. ການອອກແບບຄວນມີລັກສະນະເປັນ Module
 3. ການອອກແບບຄວນນຳສະເໜີດ້ານຂໍ້ມູນ, ສະຖາປັດຕະຍະກຳ, ພາກສ່ວນສື່ສານ ແລະ ສ່ວນປະກອບຢ່າງຊັດເຈນ
 4. ຄວນອອກແບບແຕ່ລະສ່ວນປະກອບໃຫ້ມີອິດສະຫລະຕໍ່ກັນ
 5. ຄວນອອກແບບໃຫ້ມີສ່ວນປະສານງານລະຫວ່າງສ່ວນປະກອບກັບສະພາບແວດລ້ອມພາຍນອກ
 6. ການອອກແບບຄວນເອົາຂໍ້ມູນມາຈາກການວິເຄາະລະບົບ ແລະ ໃຊ້ວິທີການປະຕິບັດດຽວກັນ

ຄຸນນະພາບ ແລະ ການປະເມີນຄຸນນະພາບການ ອອກແບບຊອບແວ

➤ ຫຼັກການອອກແບບຊອບແວ

- ແນວຄິດການອອກແບບເພື່ອນຳໄປສູ່ການອອກແບບທີ່ດີ
 7. ສັນຍາລັກທີ່ໃຊ້ໃນການອອກແບບຄວນສື່ຄວາມໝາຍໄດ້ຢ່າງຊັດເຈນ ແລະ ເປັນມາດຕະຖານ
 8. ການອອກແບບຄວນມີໂຄງສ້າງທີ່ດີ ເພື່ອແກ້ໄຂບັນຫາງ່າຍ ແລະ ປະຢັດຕົ້ນທຶນ
 9. ການອອກແບບໃນລະດັບອົງປະກອບມີລັກສະນະເປັນແບບ Functional Independence
 10. ບັນດາອົງປະກອບຂອງຊອບແວມີລັກສະນະການຂຶ້ນກັບກັນໜ້ອຍທີ່ສຸດ

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ການຄິດແບບນາມມະທຳ (Abstraction)

- ການຄິດແບບນາມມະທຳເປັນພື້ນຖານທາງຄວາມຄິດໃນການອອກແບບອັນໜຶ່ງທີ່ຊ່ວຍລຸດຄວາມຊັບຊ້ອນຂອງລະບົບ
 - ເມື່ອມີການພິຈາລະນາວິທີການແກ້ໄຂບັນຫາແຕ່ລະຢ່າງຈະເກີດການຄິດແບບນາມມະທຳຂຶ້ນເປັນລະດັບຕາມລະດັບຂອງການແກ້ໄຂບັນຫາ (ລະດັບສູງສຸດເປັນພາບລວມ ແລະ ການພົວພັນກັບພາຍນອກ)
- ໃນລະຫວ່າງການພິຈາລະນາບັນຫາແຕ່ລະລະດັບຈະມີການສ້າງ Abstraction ຂຶ້ນມາ 2 ລະດັບ
 - Procedural Abstraction ເປັນການສ້າງລຳດັບຂັ້ນຕອນຂອງຊຸດຄໍາສັ່ງຂອງ Function ໃດໜຶ່ງຂຶ້ນມາ ໂດຍບໍ່ໄດ້ກຳໜົດລາຍລະອຽດພາຍໃນ
 - Data Abstraction ເປັນຊື່ຂອງບັນດາ Object ຂໍ້ມູນທີ່ຢູ່ພາຍໃນ Procedural Abstraction

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ສະຖາປັດຕະຍະກຳ (Architecture)

- ເປັນໂຄງສ້າງທັງໝົດຂອງຊອບແວຣ໌ທີ່ສະແດງໃຫ້ເຫັນໂຄງສ້າງຂອງໂປຣແກຣມຍ່ອຍ (Module) ແລະ ການເຮັດວຽກຮ່ວມກັນຂອງໂປຣແກຣມຍ່ອຍເຫລົ່ານັ້ນ
- ນອກຈາກນັ້ນຍັງສະແດງໃຫ້ໂຄງສ້າງຂອງຂໍ້ມູນທີ່ຈະຖືກໃຊ້ຢູ່ໃນໂປຣແກຣມຍ່ອຍເຫລົ່ານັ້ນ
- ເປົ້າໝາຍຂອງການອອກແບບສະຖາປັດຕະຍະກຳແມ່ນເພື່ອ ໃຊ້ເປັນຂອບເຂດໃຫ້ແກ່ການອອກແບບສ່ວນປະກອບທີ່ເຫຼືອຂອງລະບົບໃຫ້ໄປຕາມທິດທາງດຽວກັນ ແລະ ຢູ່ໃນສະຖາປັດຕະຍະກຳດຽວກັນ
- ການອອກແບບໂຄງສ້າງສະຖາປັດຕະຍະກຳສາມາດເຮັດໄດ້ໂດຍໃຊ້ແບບຈຳລອງ 4 ປະເພດ: Structural Model, Framework Model, Dynamic Model, Process Model, Functional Model

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ແບບແຜນ (Pattern)

- ເປັນວິທີການແກ້ໄຂບັນຫາອັນໃດອັນໜຶ່ງທີ່ສາມາດນຳໄປໃຊ້ແກ້ໄຂບັນຫາຊະນິດດຽວກັນທີ່ເກີດຂຶ້ນຊ້າໆກັນໄດ້
- ຈະຕ້ອງອະທິບາຍໂຄງສ້າງຂອງການອອກແບບຊອບແວຣ໌ໄວ້ຢ່າງລະອຽດ ເຊັ່ນ: ຊື່ແບບແຜນ, ບັນຫາຂອງແບບແຜນ, ວິທີແກ້ບັນຫາ ແລະ ຜົນທີ່ເກີດຂຶ້ນ
- ການໃຊ້ແບບແຜນຈະເຮັດໃຫ້ການຜະລິດຊອບແວຣ໌ດຳເນີນໄດ້ຢ່າງວ່ອງໄວ

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ການແບ່ງລະບົບ (Modularity)

- ແມ່ນການແບ່ງລະບົບຫຼືຊອບແວຣ໌ອອກເປັນສ່ວນຍ່ອຍໆ ແຕ່ລະສ່ວນເອີ້ນວ່າ Module ຊຶ່ງຈະປະສານງານກັນເຮັດວຽກຢ່າງໃດໜຶ່ງ
- ການແບ່ງສ່ວນຈະເຮັດໃຫ້ຈັດການກັບບັນຫາແຕ່ລະສ່ວນໄດ້ງ່າຍເຊັ່ນ: ການວ່າງແຜນການພັດທະນາ, ການແກ້ໄຂຫຼືປ່ຽນແປງ, ການທົດສອບແລະ ການບໍາລຸງຮັກສາ
- ຖ້າເຫັນວ່າຍັງມີ Module ໃດຢັ້ງມີຄວາມຊັບຊ້ອນຢູ່ກໍໃຫ້ແບ່ງຍ່ອຍອອກໄປເລື້ອຍໆ ຊຶ່ງເອີ້ນຫຼັກການນີ້ວ່າ: Devide and Conquer
- ມີຄວາມຫຍຸ້ງຍາກໃນການໃຊ້ຂໍ້ມູນຮ່ວມກັນ

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ການປິດບັງລາຍລະອຽດ (Information Hiding)

- ເພື່ອແກ້ໄຂບັນຫາຂອງວິທີ Modularity, ນັກອອກແບບໄດ້ກຳໜົດ ຫຼັກການວ່າ ໃນແຕ່ລະ Module ທີ່ແຍກອອກມານັ້ນຈະຕ້ອງຊ້ອນ ລາຍລະອຽດການເຮັດວຽກຂອງມັນໄວ້ ບໍ່ວ່າຈະເປັນ ຂັ້ນຕອນການເຮັດ ວຽກ ຫຼື ຂໍ້ມູນຂອງມັນ
- ເພື່ອປ້ອງກັນການເຂົ້າໃຊ້ຂໍ້ມູນພາຍໃນ Module ໂດຍບໍ່ຈຳເປັນ ທີ່ອາດ ຈະເຮັດໃຫ້ເກີດຂໍ້ຜິດພາດໄດ້

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ຄວາມເປັນອິດສະຫລະຕໍ່ກັນໃນການເຮັດວຽກ (Functional Independence)

- ເປັນການອອກແບບຊອບແວຣ໌ໂດຍໃຫ້ແຕ່ລະ Module ປະກອບດ້ວຍ sub-function ພຽງ function ດຽວ
- ແຕ່ລະ Module ຈະຕ້ອງມີສ່ວນປະສານງານທີ່ງ່າຍ ເຮັດໃຫ້ແຕ່ລະ Module ມີຄວາມເປັນອິດສະຫລະຕໍ່ກັນ ເຮັດໃຫ້ການບໍາລຸງຮັກສາ, ການທົດສອບຊອບແວຣ໌ ງ່າຍຂຶ້ນ
- ສາມາດປະເມີນຄວາມເປັນອິດສະຫລະໄດ້ຈາກ 2 ເງື່ອນໄຂ
 - Coupling ເປັນການວັດແທກຄວາມສໍາພັນລຫວ່າງ Module (LC)
 - Cohesion ເປັນການວັດແທກລະດັບການຂຶ້ນຕໍ່ກັນຂອງໜ້າທີ່ ຫຼື ກິດຈະກຳໃນ Module (HC)

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌

➤ ການກັ່ນກອງ(Refinement)

- ເປັນຈຸດປະສົງຫຼັກຂອງວິທີການອອກແບບ Top down Design ຊຶ່ງເປັນການພັດທະນາໂປຣແກຣມໂດຍການກັ່ນກອງລາຍລະອຽດການເຮັດວຽກຂອງແຕ່ລະ Procedure ຕາມລຳດັບຂຶ້ນຕອນ
- ການອະທິບາຍລາຍລະອຽດຂອງແຕ່ລະ function ຈະເລີ່ມຕົ້ນຈາກ ຊື່ function ທີ່ຖືກກຳໜົດຂຶ້ນໃນລະດັບເທິງສຸດທີ່ເປັນແບບນາມມະທຳ ທີ່ຍັງບໍ່ທັນມີລາຍລະອຽດການເຮັດວຽກຂອງ function
- ການກັ່ນກອງແມ່ນການເອົາຊື່ຂອງ function ເລົ່ານັ້ນມາເພີ່ມເຕີມລາຍລະອຽດການເຮັດວຽກພາຍໃນໃນແຕ່ລະລະດັບ

ແນວຄິດໃນການອອກແບບຊອບແວຮ໌



➤ ການປັບໂຄງສ້າງການອອກແບບ (Refactoring)

- ເປັນເທັກນິກການປັບໂຄງສ້າງການອອກແບບພາຍໃນຂອງແຕ່ລະອົງປະກອບໂດຍບໍ່ຕ້ອງປ່ຽນ function ຫຼື ພຶດຕິກຳຂອງມັນ
- ໂດຍການເລີ່ມຕົ້ນຈາກການເອົາການອອກແບບເກົ່າມາພິຈາລະນາວ່າມີຄວາມຊ້ຳຊ້ອນບໍ່, ສ່ວນປະກອບທີ່ບໍ່ໄດ້ໃຊ້ງານມີບໍ່, ຂັ້ນຕອນການເຮັດວຽກທີ່ບໍ່ໄດ້ປະສິດທິພາບມີບໍ່, ໂຄງສ້າງຂໍ້ມູນທີ່ບໍ່ເໝາະສົມຫຼື ຜິພາດມີບໍ່

ແນວຄິດໃນການອອກແບບຊອບແວຣ໌


➤ ອອກແບບເປັນກຸ່ມ (Design Class)

- ໃນການອອກແບບຊອບແວຣ໌ຈະຕ້ອງເອົາແບບຈຳລອງ Class ທີ່ໄດ້ຈາກຂັ້ນຕອນການວິເຄາະມາກັນກອງເພື່ອກຳໜົດລາຍລະອຽດຂອງແຕ່ລະ Class
- ແລະຕ້ອງສ້າງແບບຈຳລອງ Class ທີ່ສະແດງໃຫ້ເຫັນເຖິງໂຄງສ້າງພາຍໃນທີ່ສະໜັບສະໜູນຂະບວນການທາງທຸລະກິດ ສິ່ງທີ່ໄດ້ແມ່ນ Design Class ຊຶ່ງປະກອບດ້ວຍ Design Class 5 ຊະນິດຄື: User Interface Class, Business Domain Class, Process Class, Persistent Class, ແລະ System Class

ຍຸດທະສາດ ແລະ ຫລັກການຂອງການອອກແບບ ຊອບແວຣ໌

- ຍຸດທະສາດທົ່ວໄປໃນການອອກແບບຊອບແວຣ໌ (General Strategy)
 - ປະກອບດ້ວຍ Divide-and-Conquer, Stepwise Refinement, Top-down and Bottom-up Strategy, Data Abstraction and Information Hiding, Heuristic, Design Pattern
- ການອອກແບບໃນລັກສະນະ Function (Function-oriented Design)
 - ເປັນການອອກແບບແບບ Structure ຊຶ່ງເປັນການພິຈາລະນາ Function ຂອງຊອບແວຣ໌ເປັນເກັ່ນໃນການແບ່ງຊອບແວຣ໌ອອກເປັນສ່ວນຍ່ອຍຈາກນັ້ນຈະກຳໜົດລາຍລະອຽດໃນແຕ່ລະສ່ວນຍ່ອຍ ແລະ ປັບປຸງໃນລັກສະນະໂຄງສ້າງຈາກເທິງລົງລຸ່ມ
 - ສິ່ງທີ່ໃຊ້ໃນການອອກແບບລັກສະນະໂຄງສ້າງແມ່ນ DFD, Process Description, Decision Table, Structure English, Tree Decision

ຍຸດທະສາດ ແລະ ຫຼັກການຂອງການອອກແບບ ຊອບແວ



- ການອອກແບບໃນລັກສະນະວັດຖຸ (Object-oriented Design)
 - ວິທີການອອກແບບລັກສະນະວັດຖຸເປັນການພິຈາລະນາວັດຖຸໃນ Domain ທີ່ສົນໃຈ ຖ້າເປັນຄໍານາມແມ່ນ Object, ຖ້າເປັນກິລິຍາ ແມ່ນ Method ແລະ ຄໍາຄຸນສັບແມ່ນ Attribute
 - ຈັດໂຄງສ້າງຂອງ Object ແບບ Inheritance ແລະ Polymorphism

ຍຸດທະສາດ ແລະ ຫຼັກການຂອງການອອກແບບ ຊອບແວຣ໌

- ການອອກແບບໂດຍໃຊ້ຂໍ້ມູນເປັນໃຈກາງ (Data-structure Centered Design)
 - ເປັນວິທີອອກແບບໂດຍໃຊ້ຂໍ້ມູນທີ່ Function ຈະເອົາມາປະມວນຜົນເປັນຫຼັກ
 - ເລີ່ມຕົ້ນຈາກການສະແດງໂຄງສ້າງຂໍ້ມູນໂດຍສະແດງເປັນແຜນພາບຈຳກລອງໂຄງສ້າງຂໍ້ມູນເຫຼົ່ານັ້ນ ຈາກນັ້ນທີມງານຈະເອົາແຜນພາບດັ່ງກ່າວໄປອອກແບບໂຄງສ້າງຄວບຄຸມການເຮັດວຽກຂອງໂປຣແກຣມ
- ການອອກແບບອີງປະກອບ (Component-based Design)
 - ເປັນວິທີອອກແບບຊອບແວຣ໌ດ້ວຍການແບ່ງເປັນສ່ວນປະກອບຍ່ອຍເອີ້ນວ່າ Component,
 - ແຕ່ລະ Component ຈະເຮັດວຽກເປັນ ອິດສະຫລະຕໍ່ກັນ, ເຮັດວຽກດ້ວຍຕົວເອງ ແລະ ສາມາດປະກອບເຂົ້າກັບ Component ອື່ນໆເພື່ອໃຫ້ເປັນການເຮັດວຽກຂອງຊອບແວຣ໌
 - ບັນດາ Component ຈະມີການສື່ສານກັນຜ່ານທາງ Interface
 - ວິທີດັ່ງກ່າວໄດ້ຖືກພັດທະນາຂຶ້ນມາເພື່ອຕອບສະໜອງການຜະລິດຊອບແວຣ໌ທີ່ສາມາດນຳກັບມາໃຊ້ໃໝ່ໄດ້

ແບບຈຳລອງທີ່ໃຊ້ໃນການອອກແບບ

↳ Structural Description (Static View)

- Architecture Description Language - ໃຊ້ເພື່ອອະທິບາຍສະຖາປັດຕະຍະກຳຊອບແວຮ໌ແບບຄອມໂພເນັ້ນ ແລະ ການເຊື່ອມຕໍ່ຄອມໂພເນັ້ນ
- Class and Object Diagram - ສະແດງໂຄງສ້າງຂອງ Class/Object ແລະ ຄວາມສຳພັນ
- Component Diagram - ເປັນແຜນພາບທີ່ສະແດງຄອມໂພເນັ້ນທີ່ເປັນສ່ວນປະກອບຂອງລະບົບ ແລະ ຄວາມສຳພັນ ນອກຈາກນັ້ນຍັງສະແດງໃຫ້ເຫັນ Interface ຂອງຄອມໂພເນັ້ນ
- Collaboration Responsibility Card - ໃຊ້ບັນທຶກຊື່ຄອມໂພເນັ້ນ (ຄລາດ) ພ້ອມກັບຄອມໂພເນັ້ນທີ່ມີຄວາມສຳພັນກັນ ແລະ ໜ້າທີ່ຂອງມັນ

ແບບຈຳລອງທີ່ໃຊ້ໃນການອອກແບບ

➤ Structural Description (Static View)

- Deployment Diagram - ເປັນແຜນພາບສະແດງໂຄງສ້າງທາງດ້ານ Hardware (ເອີ້ນວ່າ ໂໜດ) ຂອງລະບົບ ແລະ ຄວາມສຳພັນລະຫວ່າງໂໜດຊະນິດຕ່າງໆ
- Entity Relationship Diagram - ເປັນແຜນພາບສະແດງຄວາມສຳພັນລະຫວ່າງ Entity ໃຊ້ສະແດງໂຄງສ້າງຖານຂໍ້ມູນ
- Interface Description Language - ມີລັກສະນະຄ້າຍຄືກັບການຂຽນຄຳສັ່ງໃນໂປຣແກຣມ ໃຊ້ກຳໜົດລາຍລະອຽດຂອງ Interface ຂອງຄອມໂພເນັ້ນ ເຊັ່ນ: ຊື່, ຊະນິດ, ແລະ ການບໍລິການ
- Jackson Structure Diagram - ເປັນແຜນພາບສະແດງໂຄງສ້າງຄວບຄຸມການປະມວນຜົນຂໍ້ມູນແບບລຽງລຳດັບ, ແບບເລືອກເຮັດ ແລະ ແບບວົນຊໍ້າ
- Structure Chart - ເປັນແຜນພາບສະແດງໂຄງສ້າງຂອງໂປຣແກຣມ ສະແດງໃຫ້ເຫັນການເອິ້ນໃຊ້ Module

ແບບຈຳລອງທີ່ໃຊ້ໃນການອອກແບບ

↳ Behavioral Description (Dynamic View)

- Activity Diagram - ເປັນແຜນພາບສະແດງລຳດັບການປະຕິບັດກິດຈະກຳຂອງລະບົບທີ່ເກີດຈາກການທຳງານຂອງ Object
- Collaborative Diagram - ເປັນແຜນພາບສະແດງໃຫ້ເຖິງການພົວພັນກັນລະຫວ່າງ Object
- Data Flow Diagram - ເປັນແຜນພາບການໄຫຼຂອງຂໍ້ມູນຈາກຂະບວນການໜຶ່ງຫາອີກຂະບວນການໜຶ່ງ ຫຼື ພາກສ່ວນອື່ນໆ
- Decision Table and Diagram - ເປັນຕາຕະລາງຕັດສິນໃຈ ຊຶ່ງສະແດງໃຫ້ເຫັນການຕັດສິນໃຈປະຕິບັດກິດຈະກຳຢ່າງໃດໜຶ່ງຂອງລະບົບ
- Flowchart and Structured Flowchart - ເປັນແຜນພາບສະແດງລຳດັບການປະຕິບັດກິດຈະກຳ

ແບບຈຳລອງທີ່ໃຊ້ໃນການອອກແບບ

↳ Behavioral Description (Dynamic View)

- Sequence Diagram - ສະແດງການພົວພັນກັນລຫວ່າງ Object ຕາມລຳດັບເວລາ
- State Transition and Statechart Diagram - ເປັນແຜນພາບທີ່ສະແດງເຖິງພຶດຕິກຳຂອງ Object ໂດຍສະແດງເຖິງສະຖານະພາບ ແລະ ການປ່ຽນສະຖານະພາບຂອງ Object ທີ່ມີຕໍ່ເຫດການໃດເຫດການໜຶ່ງ
- Formal Specification Language - ໃຊ້ກຳໜົດລາຍລະອຽດຂອງ Interface ແລະ ພຶດຕິກຳຂອງຄອມໂພເນັ້ນ
- Pseudo-Code and Program Design Language - ມີລັກສະນະຄ້າຍຄືກັບການຂຽນຄຳສັ່ງໃນໂປຣແກຣມ ຊຶ່ງເອີ້ນວ່າລະຫັດທຽມ ໃຊ້ຈຳລອງການເຮັດວຽກຂອງ Function