System Analysis and Design

ໂດຍ: ອຈ ສົມມິດ ທຸມມາລີ

Email: mithcom@yahoo.com

s.thoummaly@nuol.edu.la

Mobile: 020 55720450

ບົດທີ 2

ການພັດທະນາລະບົບຂ່າວສານ
(Information System Development)

ຈຸດປະສົງ

- ສາມາດລະບຸຂັ້ນຕອນ ແລະ ລາຍລະອຽດຂອງວົງຈອນພັດທະນາລະບົບແຕ່ລະໄລຍະໄດ້.
- ເຂົ້າໃຈລະບຽບວິທີ, ແບບຈຳລອງ, ເຄື່ອງມື ແລະ ເຕັກນິກຕ່າງໆທີ່ນຳມາໃຊ້ກັບຂະບວນການການພັດທະນາລະບົບ.
- ບອກຂໍ້ແຕກຕ່າງລະຫວ່າງການພັດທະນາລະບົບແບບໂຄງສ້າງ ແລະ ວິທີພັດທະນາລະບົບແບບວັດຖຸໄດ້.
- ເຫັນຄວາມສຳຄັນຂອງວິສະວະກຳຊອບແວທີ່ມຸ້ງເນັ້ນເຖິງການພັດທະນາຊອບແວໃຫ້ມີຄຸນນະພາບ.
- ສາມາດນຳເອົາ Model ການພັດທະນາຊອບແວມາປະຫຍຸກໄດ້ກັບການພັດທະນາລະບົບອຽກໄດ້ຢ່າງເໝາະສົມ
- ບອກຄຸນສົມບັດຂອງຊອບແວທີ່ມີຄຸນນະພາບໄດ້.

ຫົວຂໍ້ສອນ

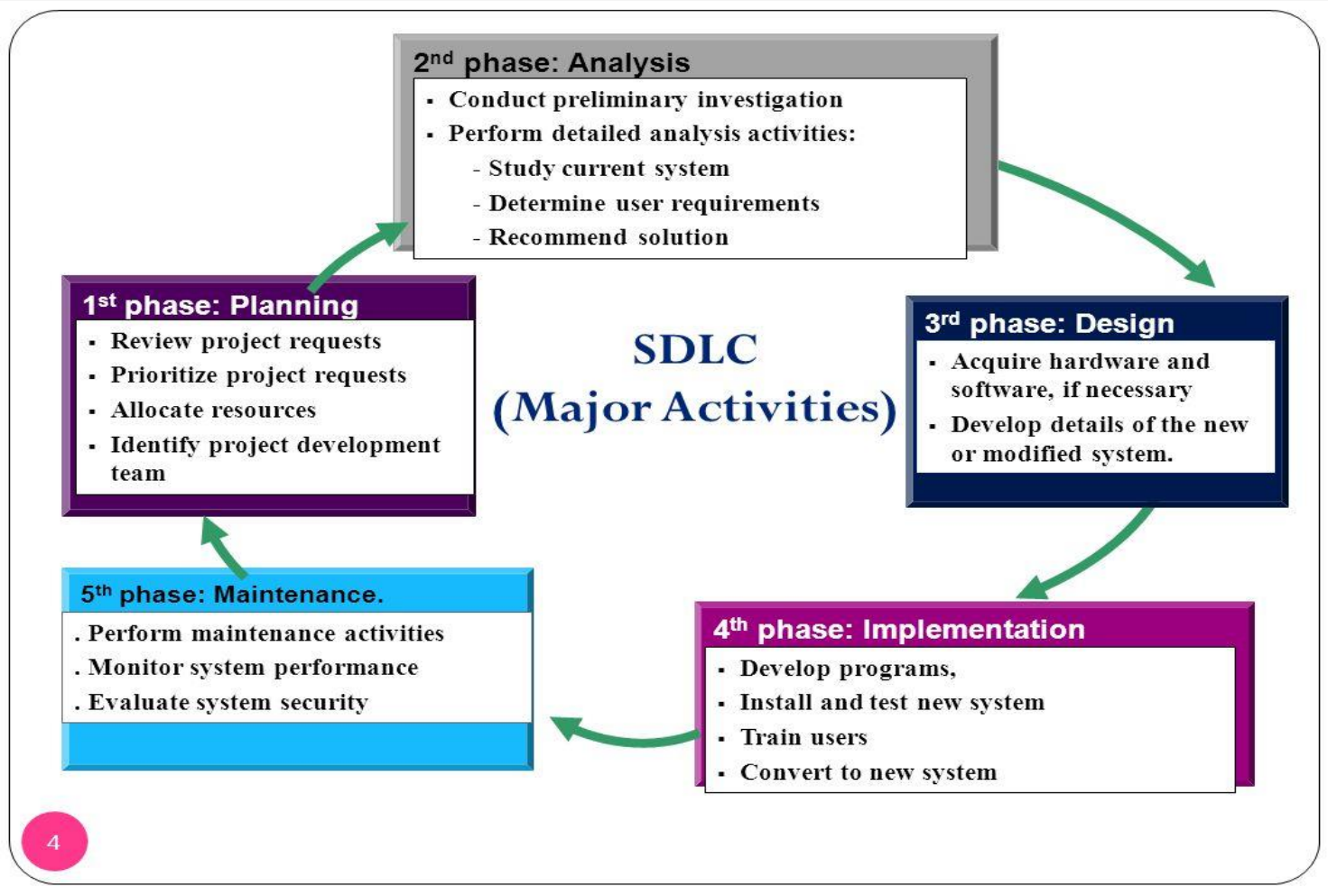
- ວົງຈອນການພັດທະນາລະບົບ
- ລະບຽບວິທີ, ແບບຈຳລອງ, ເຄື່ອງມື ແລະ ເຕັກນິກ
- ວິທີການພັດທະນາລະບົບ
- ວິສະວະກຳຊອບແວ
- ຮູບແບບການພັດທະນາຊອບແວ
- ມາດຕະຖານຄຸນນະພາບຂອງລະບົບຂ່າວສານ
- ເຄື່ອງມືທີ່ໃຊ້ສະໜັບໜູນການພັດທະນາລະບົບ

ວົງຈອນການພັດທະນາລະບົບ

ວົງຈອນການພັດທະນາລະບົບ (System Development Life Cycle: SDLC) ເປັນຂະບວນການທີ່ສະແດງເຖິງກິດຈະກຳຕ່າງໆໃນແຕ່ລະຂັ້ນຕອນການເຮັດວຽກຂອງລະບົບຕັ້ງແຕ່ເລີ່ມຕົ້ນຈົນຈົບ.

SDLC ມີຂອບເຂດການເຮັດວຽກທີ່ມີໂຄງສ້າງຢ່າງຊັດເຈນ ແລະ ມີການຈັດລຳດັບກິດຈະກຳແຕ່ລະໄລຍະທີ່ແນ່ນອນ ເຊັ່ນວ່າ ເມື່ອສຳເລັດໄລຍະການວິເຄາະແລ້ວ ຂັ້ນຕອນຕໍ່ໄປກໍຄືການອອກແບບ ເປັນຕົ້ນ, ດັ່ງນັ້ນ ຈຶ່ງເຮັດໃຫ້ເຂົ້າໃຈເຖິງກິດຈະກຳພື້ນຖານ, ຂອບເຂດ ແລະ ລາຍລະອຽດຕ່າງໆໃນແຕ່ລະໄລຍະຂອງການພັດທະນາລະບົບ.

ແຜນວາດວົງຈອນການພັດທະນາລະບົບ



ວົງຈອນການພັດທະນາລະບົບ

1. ໄລຍະທີ 1: ການວາງແຜນໂຄງການ
2. ໄລຍະທີ 2: ການວິເຄາະ
3. ໄລຍະທີ 3: ການອອກແບບ
4. ໄລຍະທີ 4: ການນຳໄປໃຊ້
   - ການພັດທະນາ
   - ການທົດສອບ
   - ການຕິດຕັ້ງ
5. ໄລຍະທີ 5: ການນຳບາລຸງຮັກສາ

ໄລຍະທີ 1: ການວາງແຜນໂຄງການ

ການວາງແຜນໂຄງການເປັນຂະບວນການພື້ນຖານຂອງຄວາມເຂົ້າໃຈວ່າ ເປັນຫຍັງ ຈຶ່ງສ້າງລະບົບຂຶ້ນມາ ແລະ ຕ້ອງກຳນົດທິມງານເພື່ອມາດຳເນີນການສ້າງລະບົບນີ້ໄດ້ແນວໃດ, ນອກຈາກນີ້ຍັງໄດ້ຄຳນຶງເຖິງຄວາມຄຸ້ມຄ່າກັບການລົງທຶນວ່າເໝາະສົມ ຫຼື ບໍ່.

ສິ່ງທີ່ຕ້ອງດຳເນີນໃນ ໄລະຍະທີ 1

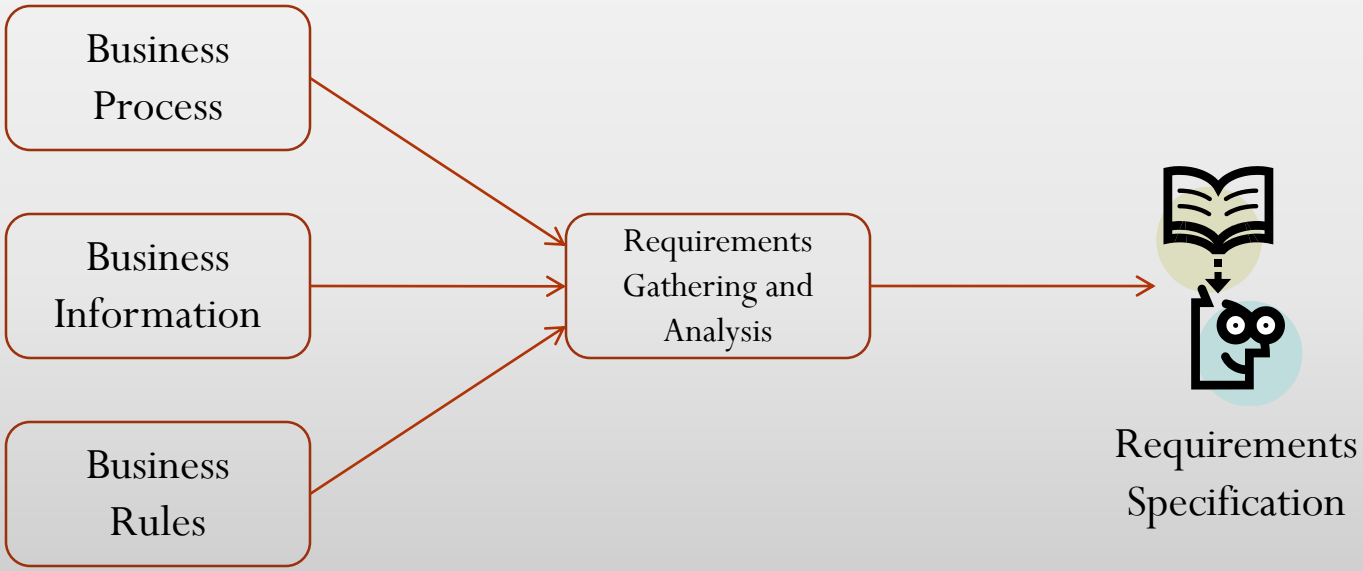
- ກຳນົດບັນຫາ
- ກຳນົດເອລາໂຄງການ
- ສຶກສາຄວາມເປັນໄປໄດ້ຂອງໂຄງການ
- ຈັດຕັ້ງທີມງານໂຄງການ
- ດຳເນີນໂຄງການ

ໄລະຍະທີ 2: ການວິເຄາະ

ໄລຍະການວິເຄາະຈະຕ້ອງຕອບຄຳຖາມ ຄື: ໃຜເປັນຜູ້ໃຊ້ລະບົບ, ມີຫຍັງແດ່ທີ່ຕ້ອງເຮັດ ແລະ ເຮັດຢູ່ໃສເມື່ອໃດ ໂດຍໄລຍະນີ້ທີມງານຈະສຶກສາລະບົບງານປະຈຸບັນພ້ອມທັງລະບຸແນວທາງໃນການປັບປຸງຂະບວນການໃຫ້ດີຂຶ້ນ ເພື່ອພັດທະນາແນວຄິດສຳລັບລະບົບໃໝ່ຂຶ້ນມາ.

ສິ່ງສຳຄັນຂອງໄລຍະນີ້ກໍຄື ການຮວບຮວມຄວາມຕ້ອງການ ໂດຍສາມາດຮວບຮວມໄດ້ຈາກການສັງເກດ, ການສຳພາດ, ການສ້າງແບບສອບຖາມ ລວມໄປເຖິງການອ່ານເອກະສານກ່ຽວກັບການປະຕິບັດງານຂອງລະບົບປະຈຸບັນ ພ້ອມທັງລະບຸບການຕ່າງໆຂອງອົງກອນ.

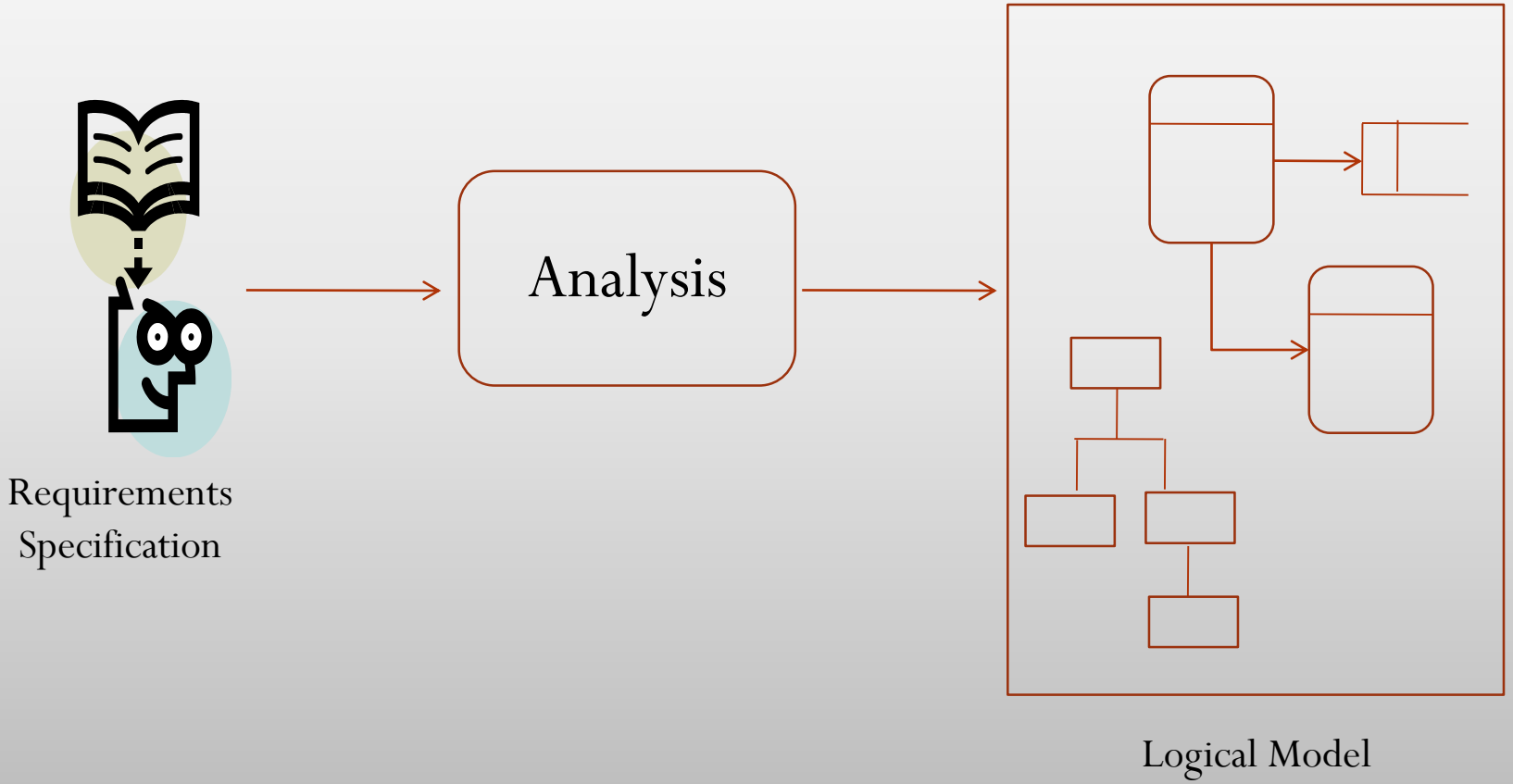
ເກັບກຳຄວາມຕ້ອງການເພື່ອສະຫຼຸບເປັນຂໍ້ກຳນົດ



ຂໍ້ສະເໜີລະບົບ (System Proposal)

ຫຼັງຈາກໄດ້ຄວາມຕ້ອງການໃນດ້ານຕ່າງໆແລ້ວ ຈຶ່ງນຳມາເປັນຂໍ້ກຳນົດທີ່ຊັດເຈນ ແລະ ຂັ້ນຕອນຕໍ່ໄປກໍຄືນຳເອົາແນວຄິດກ່ຽວກັບລະບົບພ້ອມທັງແບບຈຳລອງມາລວມເຂົ້າກັນເປັນເອກະສານທີ່ເອີ້ນວ່າ ຂໍ້ສະເໜີລະບົບ (System Proposal) ເພື່ອນຳສະເໜີຕໍ່ເຈົ້າຂອງໂຄງການວ່າຈະໃຫ້ມີການດຳເນີນການຫຼືບໍ່, ໂດຍເອກະສານດັ່ງກ່າວຈະປະກອບດ້ວຍລາຍລະອຽດຄວາມຕ້ອງການຂອງລະບົບໃໝ່ທີ່ຖືກນຳສະເໜີຜ່ານແບບຈຳລອງຂະບວນການ ພ້ອມທັງ ແບບຈຳລອງຂໍ້ມູນ ເປັນຕົ້ນ.

ຂັ້ນຕອນການນຳເອົາຂໍ້ກຳນົດມາວິເຄາະໃນລາຍລະອຽດເພື່ອສ້າງເປັນແບບຈຳລອງຂອງລະບົບໃໝ່ (Logical Model)



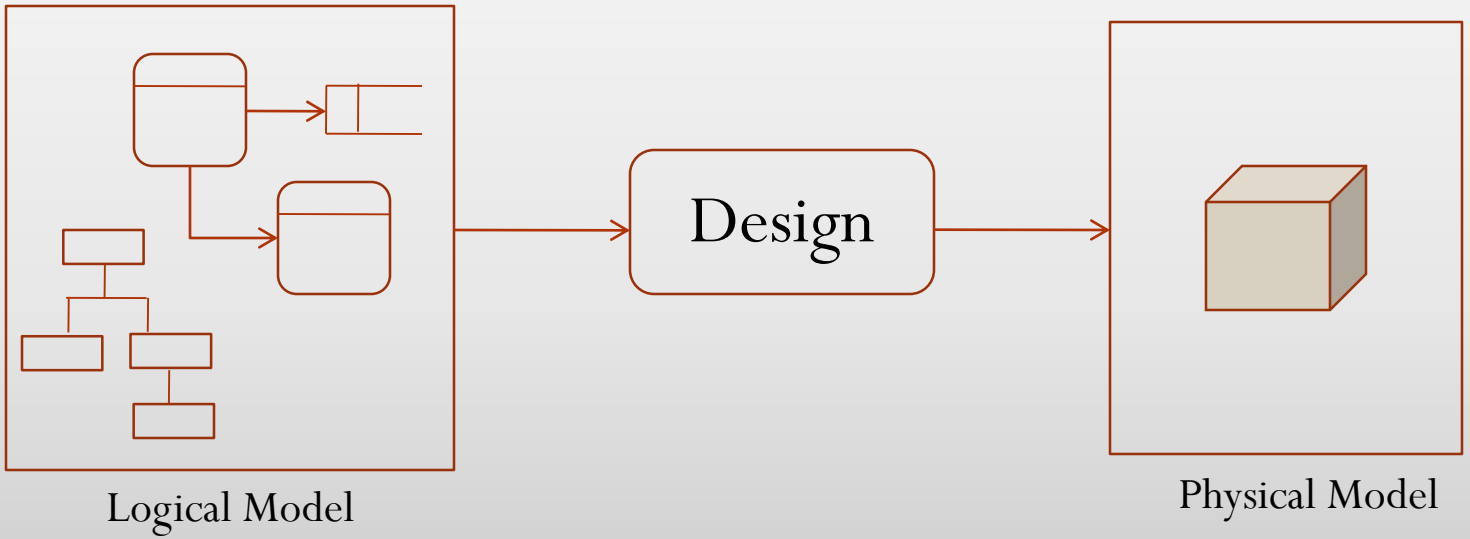
ສິ່ງທີ່ຕ້ອງດຳເນີນໃນ ໄລະຍະທີ 2

- ວິເຄາະລະບົບງານປະຈຸບັນ
- ຮວບຮວມຄວາມຕ້ອງການໃນດ້ານຕ່າງໆ ແລະ ນຳມາວິເຄາະເພື່ອ ສະຫຼຸບເປັນຂໍ້ກຳນົດທີ່ຊັດເຈນ
- ນຳຂໍ້ກຳນົດພັດທະນາອອກມາເປັນຄວາມຕ້ອງການລະບົບໃໝ່
- ສ້າງແບບຈຳລອງຂະບວນການຂອງລະບົບໃໝ່ໂດຍການສ້າງແຜນ ວາດການໄຫຼຂໍ້ມູນໃນລະດັບຕ່າງໆ
- ສ້າງແບບຈຳລອງຂໍ້ມູນດ້ວຍ ER Diagram

ໄລະຍະທີ 3: ການອອກແບບ

ໃນໄລຍະນີ້ເປັນການຕັດສິນໃຈວ່າ ລະບົບຈະດຳເນີນການໄປໄດ້ແບບໃດ ເຊັ່ນວ່າ ການຈັດຫາອຸປະກອນ Hardware, Software, ໂຄງສ້າງຂອງເຄືອຄ່າຍທີ່ຈະນຳມາໃຊ້, ການຕິດຕໍ່ສື່ສານລະຫວ່າງຜູ້ໃຊ້ກັບລະບົບ ລວມໄປເຖິງແບບຟອມ ແລະ ລາຍງານຕ່າງໆ ນອກຈານີ້ຍັງຈະຕ້ອງຄຳນຶງເຖິງໂປຣແກຣມຖານຂໍ້ມູນ ແລະ ແຟ້ມຂໍ້ມູນທີ່ຈຳເປັນ.

ຂັ້ນຕອນການນຳເອົາແບບຈຳລອງ Logical Model ມາຜ່ານການອອກແບບ ເພື່ອພັດທະນາເປັນແບບຈຳລອງ Physical Model



ການອອກແບບ

ໄລຍະນີ້ **ກົນລະຍຸດໃນການຈັດຫາລະບົບ** ຈະຕ້ອງໄດ້ຮັບການພັດທະນາຂຶ້ນມາເປັນອັນດັບທຳອິດເພື່ອຈະໄດ້ສ້າງຄວາມຈະແຈ້ງກ່ຽວກັບແນວທາງໃນການພັດທະນາລະບົບ ວ່າຈະເລືອກແນວທາງໃດ ເຊັ່ນ: ຊື້ໂປຣແກຣມສຳເລັດຮູບ, ພັດທະນາຂຶ້ນມາໃຊ້ເອງ ຫຼື ຈ້າງບໍລິສັດມາພັດທະນາໃຫ້ ຈາກນັ້ນ ກໍທຳການ **ອອກແບບສະຖາປັດຍະກຳຂອງລະບົບ** ທີ່ອະທິບາຍເຖິງ Hardware, Software ແລະ ໂຄງສ້າງດ້ານເຄືອຂ່າຍ ເຊິ່ງສ່ວນໃຫຍ່ແລ້ວຈະປັບປຸງລະບົບເດີມທີ່ຢູ່ໃນອົງກອນນັ້ນໆ **ສ່ວນ ການອອກແບບ Interface** ຈະກ່ຽວຂ້ອງກັບການສື່ສານລະຫວ່າງຜູ້ໃຊ້ ກັບລະບົບ ເຊັ່ນວ່າ ການໂຕ້ຕອບກັນຜ່ານເມນູ, ປຸ່ມຕ່າງໆ ເທິງໜ້າຈໍ ລວມໄປເຖິງແບບຟອມ ແລະ ລາຍງານທີ່ລະບົບຕ້ອງໃຊ້. ສ່ວນ **ການອອກແບບຖານຂໍ້ມູນ** ກໍຈະຖືກພັດທະນາຂຶ້ນມາເພື່ອໃຫ້ຮູ້ວ່າມີຂໍ້ມູນຫຍັງແດ່ທີ່ຕ້ອງຈັດເກັບໃນຖານຂໍ້ມູນ ແລະ ສຸດທ້າຍກໍຄື **ການອອກແບບໂປຣແກຣມ** ດ້ວຍພາສາຄອມພິວເຕີ ເພື່ອນຳໄປໃຊ້ໃນການຂຽນໂປຣແກຣມໃນໄລຍະການນຳໄປໃຊ້(ໄລຍະທີ 4).

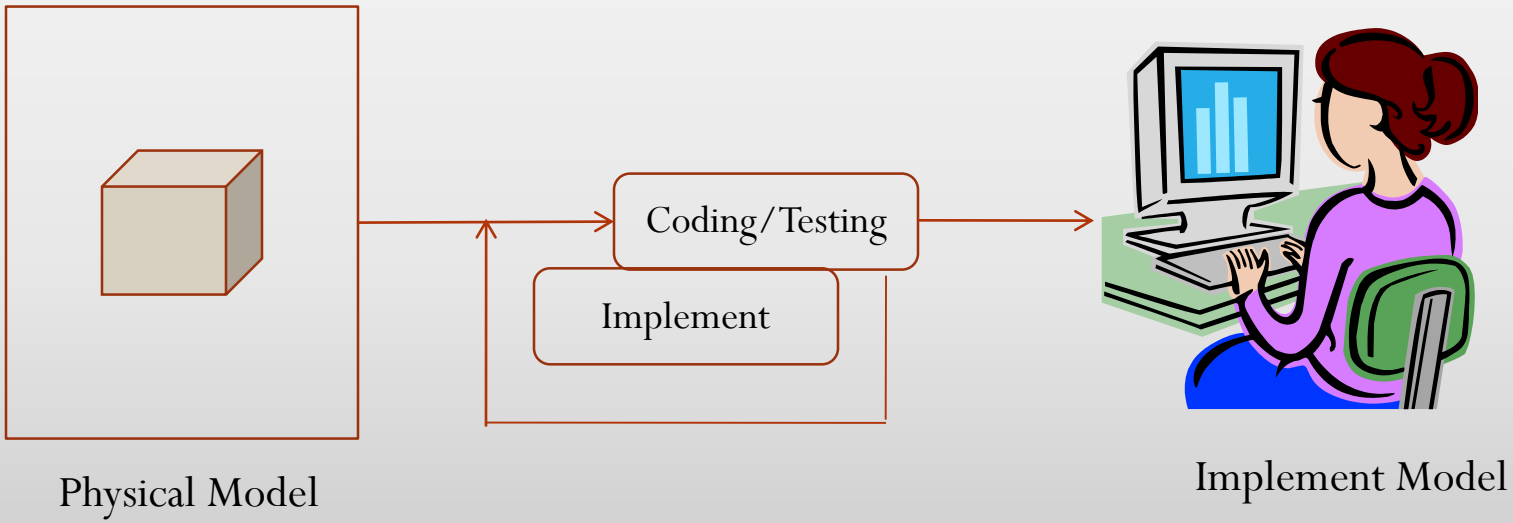
ສິ່ງທີ່ຕ້ອງດຳເນີນໃນ ໄລະຍະທີ 3

- ການຈັດຫາລະບົບ
- ອອກແບບສະຖາປັດຍະກຳຂອງລະບົບ (Architecture Design)
- ອອກແບບຖານຂໍ້ມູນ (Database Design)
- ອອກແບບຟອມລາຍງານ (Output Design)
- ອອກແບບຟອມປ້ອນຂໍ້ມູນ (Input Design)
- ອອກແບບ User Interface (User Interface Design)
- ສ້າງຕົ້ນແບບ (Prototype)
- ອອກແບບໂປຣແກຣມ (Structure Chart)

ໄລະຍະທີ 4: ການນຳໄປໃຊ້

ໄລຍະນີ້ຈະກ່ຽວຂ້ອງກັບການສ້າງ, ການທົດສອບ ແລະ ການຕິດຕັ້ງລະບົບໂດຍມີຈຸດປະສົງຫຼັກບໍ່ພຽງແຕ່ສ້າງຜະລິດຕະພັນໃຫ້ກົງກັບຄວາມຕ້ອງການຂອງອົງກອນເທົ່ານັ້ນ ແຕ່ຈະຕ້ອງລວມໄປເຖິງຄວາມໝັ້ນໃຈວ່າຜູ້ໃຊ້ລະບົບທຸກໆຄົນຕ້ອງໄດ້ຜ່ານການຝຶກອົບຮົມການໃຊ້ງານເພື່ອກຽມຄວາມພ້ອມຕໍ່ການໃຊ້ລະບົບຂ່າວສານໃຫ້ເກີດປະໂຫຍດຕໍ່ອົງກອນຕາມຈຸດປະສົງທີ່ຕັ້ງໄວ້.

ຂຽນໂປຣແກຣມ, ທົດສອບ ແລະ ນຳໄປໃຊ້ (Coding/Testing and Implement)



ການສ້າງລະບົບ

ການສ້າງລະບົບ ເປັນຂັ້ນຕອນທຳອິດໃນໄລຍະນີ້ ໂດຍລະບົບທີ່ສ້າງຂຶ້ນມາຈະໄດ້ຮັບການທົດສອບເພື່ອໃຫ້ເກີດຄວາມໝັ້ນໃຈວ່າສາມາດດຳເນີນງານກົງຕາມທີ່ອອກແບບໄວ້ຫຼືບໍ່ ນອກຈາກນີ້ຍັງມີກິດຈະກຳ ການແປງຂໍ້ມູນ ຖືວ່າເປັນໜຶ່ງໃນກິດຈະກຳທີ່ສຳຄັນ ເນື່ອງຈາກກິດຈະກຳດັ່ງກ່າວແມ່ນການແປງຂໍ້ມູນຈາກລະບົບເດີມມາສູ່ລະບົບໃໝ່ ເມື່ອມີການຕິດຕັ້ງຈຶ່ງສາມາດພ້ອມໃຊ້ງານໄດ້ເລີຍ.

ສິ່ງທີ່ຕ້ອງດຳເນີນໃນ ໄລະຍະທີ 4

- ສ້າງລະບົບຂຶ້ນມາດ້ວຍການຂຽນໂປຣແກຣມ
- ກວດສອບຄວາມຖືກຕ້ອງ ແລະ ທົດສອບລະບົບ
- ແປງຂໍ້ມູນ
- ຕິດຕັ້ງລະບົບ ແລະ ສ້າງຄູ່ມືລະບົບ
- ຝຶກອົບຮົມໃຫ້ຜູ້ໃຊ້ ແລະ ປະເມີນຜົນລະບົບໃໝ່

ໄລະຍະທີ 5: ການນຳບາລຸງຮັກສາ

ໂດຍທົ່ວໄປແລ້ວ ໄລຍະນີ້ບໍ່ໄດ້ຖືກບັນຈຸໃນຂັ້ນຕອນ SDLC ຈົນກະທັ້ງໄດ້ຕິດຕັ້ງໃຊ້ງານແລ້ວ ເນື່ອງຈາກວ່າເປັນໄລຍະທີ່ໃຊ້ເວລາຫຼາຍທີ່ສຸດ ຖ້າທຽບກັບໄລຍະອື່ນໆ ເພາະວ່າຕ້ອງໄດ້ບຳລຸງຮັກສາໃຫ້ລະບົບສາມາດໃຊ້ງານໄດ້ຍາວນານ ແລະ ຮອງຮັບເຕັກໂນໂລຊີໃໝ່ໆໃນອານາຄົດ ດັ່ງນັ້ນ ຈຶ່ງ ສາມາດເພີ່ມເຕີມຄຸນສົມບັດໃໝ່ໆເຂົ້າໄປເພື່ອເພີ່ມປະສິດທິພາບໃນການທຳງານໃຫ້ກັບລະບົບໄດ້ ເຊິ່ງຄຸນສົມບັດດັ່ງກ່າວອາດຈະມາຈາກຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ ເຊັ່ນວ່າ ຜູ້ໃຊ້ອາດຈະເຫັນຂໍ້ບົກຜ່ອງຂອງລະບົບເມື່ອໃຊ້ງຶກໄປດົນໆເຊິ່ງ ຕ້ອງໄດ້ຮັບການແກ້ໄຂໃຫ້ຖືກຕ້ອງ, ລວມທັງຂໍຮ້ອງໃຫ້ຂຽນໂມດູນໂປຣແກຣມ ໃໝ່ເພີ່ມເຕີມ, ເພື່ອສະໜັບສະໜູນການທຳງານຂອງອົງກອນ ເປັນຕົ້ນ.

ສິ່ງທີ່ຕ້ອງດຳເນີນໃນ ໄລະຍະທີ 5

- ການບຳລຸງຮັກສາລະບົບ
- ການເພີ່ມເຕີມຄຸນສົມບັດໃໝ່ເຂົ້າໄປໃນລະບົບ
- ການສະໜັບສະໜູນວຽກຂອງຜູ້ໃຊ້

ວິທີການພັດທະນາລະບົບ (System Development Methodology)

ໃນການພັດທະນາລະບົບ, ນັກວິເຄາະລະບົບສາມາດນຳເອົາ ຮູບແບບ, ເຄື່ອງມື ແລະ ເຕັກນິກ ມາປະຍຸກໃຊ້ກັບການວິເຄາະ ແລະ ອອກແບບລະບົບໄດ້ ໂດຍເອີ້ນວ່າ Methodology. ເຊິ່ງ ສິ່ງເຫຼົ່ານີ້ຈະຊ່ວຍເຮັດໃຫ້ຜູ້ພັດທະນາລະບົບສາມາດພັດທະນາໄດ້ ໄວ ແລະ ມີປະສິດທິພາບ.

ຮູບແບບ (Model)

- ຜັງງານ (Flow chart)
- ແຜນວາດການໄຫຼຂໍ້ມູນ (Data Flow Diagram)
- ແຜນວາດຄວາມສຳພັນ (ER Diagram)
- ຜັງໂຄງສ້າງ (Structure Chart)
- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Gantt Chart/PERT
- Organizational Hierarchy Chart

ເຄື່ອງມື (Tools)

- ໂປຣແກຣມຈັດການໂຄງການ
- ໂປຣແກຣມ/ເຄື່ອງມືຊ່ວຍແຕ້ມ
- ໂປຣແກຣມປະມວນຜົນຄຳ ຫຼື ໂປຣແກຣມ Editor
- Case Tools/Visual Modeling Tools
- ໂປຣແກຣມຈັດການຖານຂໍ້ມູນ
- ໂປຣແກຣມແປງ Diagram ເປັນລະຫັດຄຳສັ່ງ

ເຕັກນິກ (Techniques)

- ເຕັກນິກການບໍລິຫານໂຄງການ
- ເຕັກນິກການສຳພາດຜູ້ໃຊ້
- ເຕັກນິກການສ້າງແບບຈຳລອງຂໍ້ມູນ
- ເຕັກນິກການອອກແບບຖານຂໍ້ມູນແບບສຳພັນ
- ເຕັກນິກການວິເຄາະແບບໂຄງສ້າງ
- ເຕັກນິກການອອກແບບໂຄງສ້າງ
- ເຕັກນິກການຂຽນໂປຣແກຣມແບບໂຄງສ້າງ
- ເຕັກນິກການທົດສອບຊອບແວ
- ເຕັກນິກການວິເຄາະ ແລະ ອອກແບບລະບົບແບບວັດຖຸ

ການພັດທະນາລະບົບ

1. ການພັດທະນາລະບົບແບບໂຄງສ້າງ

   (Structured System Development)

2. ການພັດທະນາລະບົບແບບອັດຖ

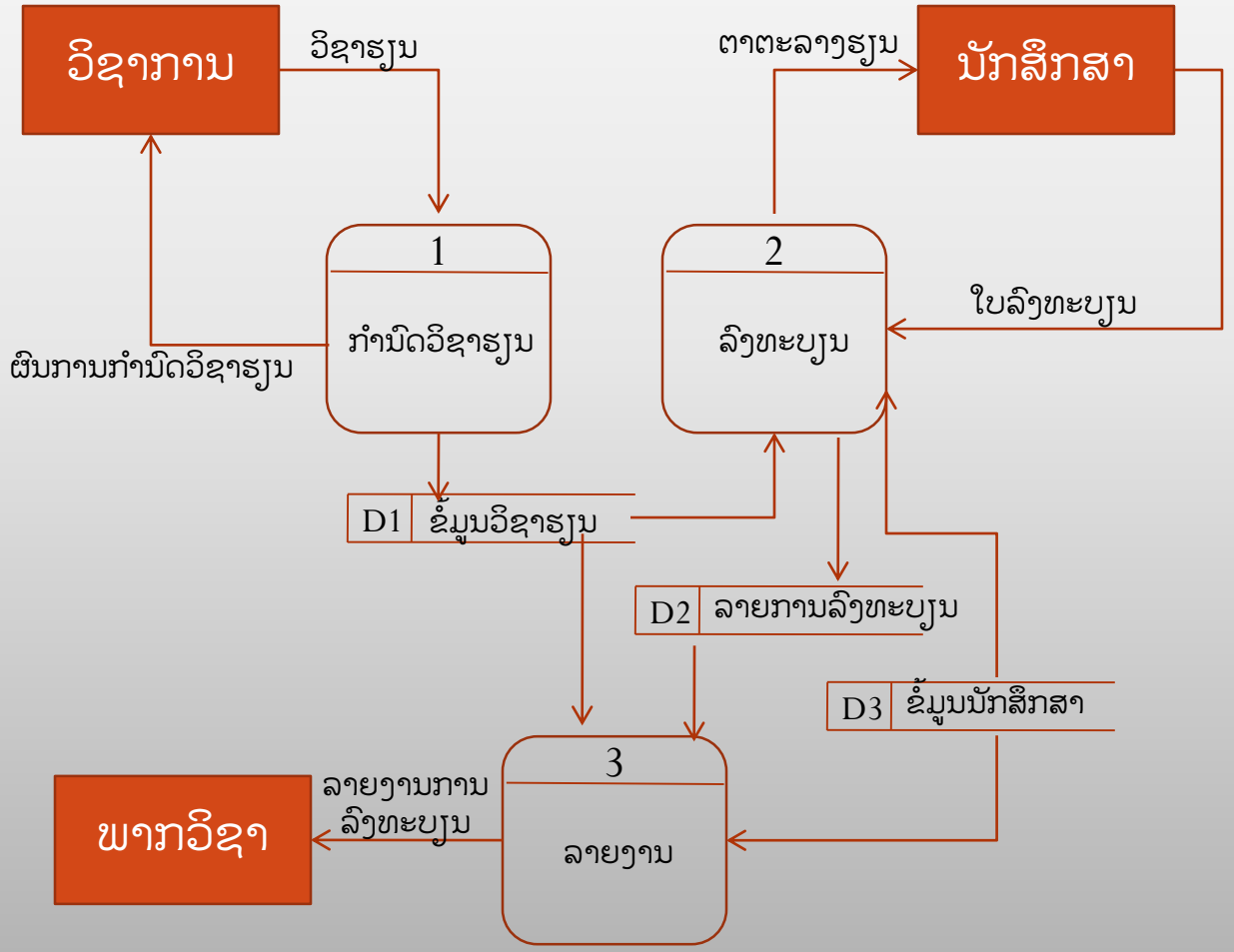
   (Object-Oriented System Development)

ການພັດທະນາລະບົບແບບແບບໂຄງສ້າງ

ເຕັກນິກການພັດທະນາລະບົບແບບເກົ່າເປັນການພັດທະນາລະບົບແບບໂຄງສ້າງເຊິ່ງປະກອບດ້ວຍ:

1. ການວິເຄາະແບບໂຄງສ້າງ
2. ການອອກແບບທາງໂຄງສ້າງ
3. ການຂຽນໂປຣແກຣມແບບໂຄງສ້າງ

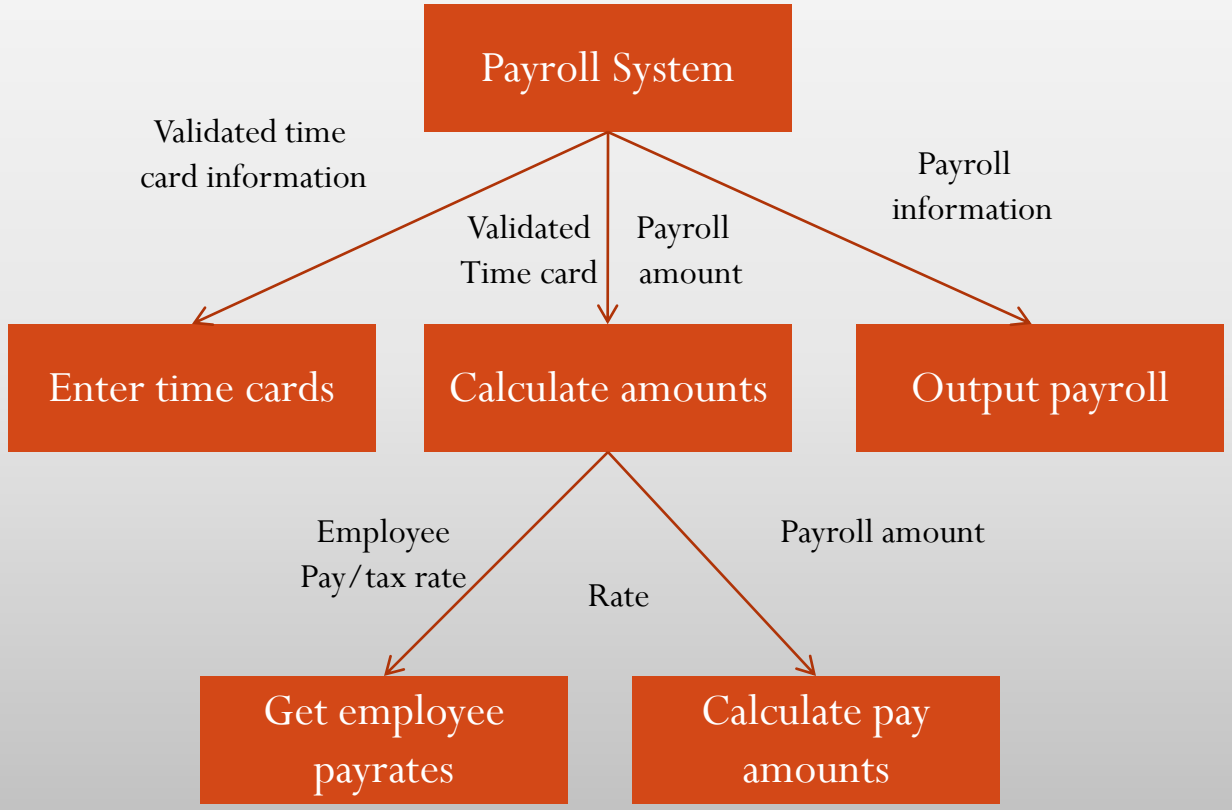
ການວິເຄາະແບບໂຄງສ້າງ (Structured Analysis)



Entity Relationship Diagram: ERD

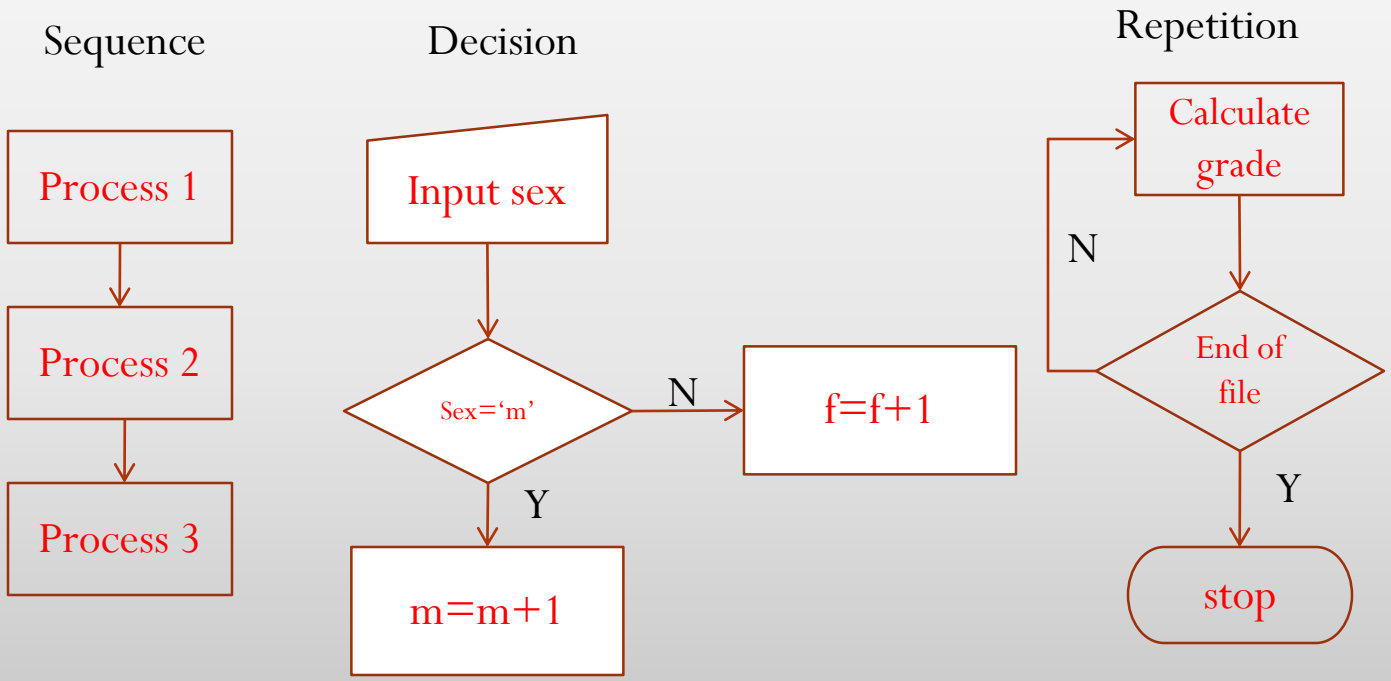
ນອກຈາກການວິເຄາະໂຄງສ້າງແລ້ວໃນເວລາດຽວກັນ ແບບຈຳລອງຂໍ້ມູນກໍໄດ້ຮັບການສ້າງຂຶ້ນ ເຊັ່ນ ຈາກຮູບເຮົາສັງເກດເຫັນວ່າ Process “ລົງທະບຽນ” ຈະກ່ຽວຂ້ອງກັບ “ຂໍ້ມູນນັກສຶກສາ”, “ຂໍ້ມູນລາຍວິຊາ” ແລະ “ຂໍ້ມູນລົງທະບຽນ” ເຊິ່ງຂໍ້ມູນເລົ່ານີ້ຈະມີຄວາມສຳພັນກັນແບບ Entity Relationship Diagram: ERD ດັ່ງນີ້

ການອອກແບບທາງໂຄງສ້າງ (Structured Design)



ການອອກແບບທາງໂຄງສ້າງເຮັດໃຫ້ເຮົາສາມາດຮູ້ວ່າ ສິ່ງທີ່ລະບົບຕ້ອງເຮັດມີຫຍັງແດ່, ແມ່ນຫຍັງຄືຟັງຊັນຫຼັກຂອງ ລະບົບ, ຂໍ້ມູນທີ່ຕ້ອງການໃຊ້ມີຫັຍງແດ່ ແລະ Output ທີ່ຕ້ອງການເປັນແນວໃດເປັນຕົ້ນ

ການຂຽນໂປຣແກຣມແບບໂຄງສ້າງ



ການພັດທະນາລະບົບແບບອັດຖ

ການພັດທະນາລະບົບແບບອັດຖຈະເບິ່ງລະບົບຂ່າວສານເປັນແຫຼງລວມ ຂອງການໂຕ້ຕອບລະຫວ່າງອັດຖເພື່ອທຳງານຮ່ວມກັນຈົນສຳເລັດ ໃນ ການພັດທະນາລະບົບແບບນີ້ ຈະບໍ່ມີຂະບວນການ ແລະ ຂໍ້ມູນທີ່ແຍກ ອອກຈາກກັນຄືການພັດທະນາແບບໂຄງສ້າງ. ໂດຍລະບົບຈະມີພຽງແຕ່ ອັດຖທີ່ເປັນສິ່ງໃດໜຶ່ງໃນລະບົບຄອມພິວເຕີທີ່ມີຄວາມສາມາດໃນການ ຕອບສະໜອງຕໍ່ຂ່າວສານ ດັ່ງນັ້ນມຸມມອງໃນການພັດທະນາລະບົບແບບ ນີ້ຈຶ່ງມີຄວາມແຕກຕ່າງຈາການພັດທະນາລະບົບແບບໂຄງສ້າງ ບໍ່ວ່າຈະ ເປັນການວິເຄາະ, ອອກແບບ ແລະ ຂຽນໂປຣແກຣມ.

ເຮັດໃບສັ່ງຊື້ນໃຫ້ກັບລູກຄ້າ ຊື່ ສົມສີ ໂຊກຊ່ວຍ ເຊິ່ງປະກອບດ້ວຍໂຕະເຮັດວຽກສຳລັບຜູ້ບໍລິຫານ ແລະ ຕັ່ງນວມ

ອັດຖຸ: New Order
ເລກທີ 134
ວັນ10/10/2016

ຕ້ອງ ແລະ ນີ້ຄື ລາຍລະອຽດຂອງໃບ ສັ່ງຊື້ໃໝ ເລກທີ 134

ເພີ່ມລາຍໂຕະເຮັດ ວຽກສຳລັບຜູ້ບໍລິຫານ ລະຫັດ 19874 ລົງໃນໃບສັ່ງຊື້ນ

ອັດຖຸ Product: ໂຕະເຮັດວຽກສຳລັບຜູ້ບໍລິຫານ
ລະຫັດສິນຄ້າ: 19874

ຕ້ອງ ຈະດຳເນີ ນການໃຫ້

ເພີ່ມລາຍການຕັ່ງ ນວມ ລະຫັດ 76532 ລົງໃນໃບສັ່ງຊື້ນ

ອັດຖຸ Product: ຕັ່ງນວມ
ລະຫັດສິນຄ້າ: 76532

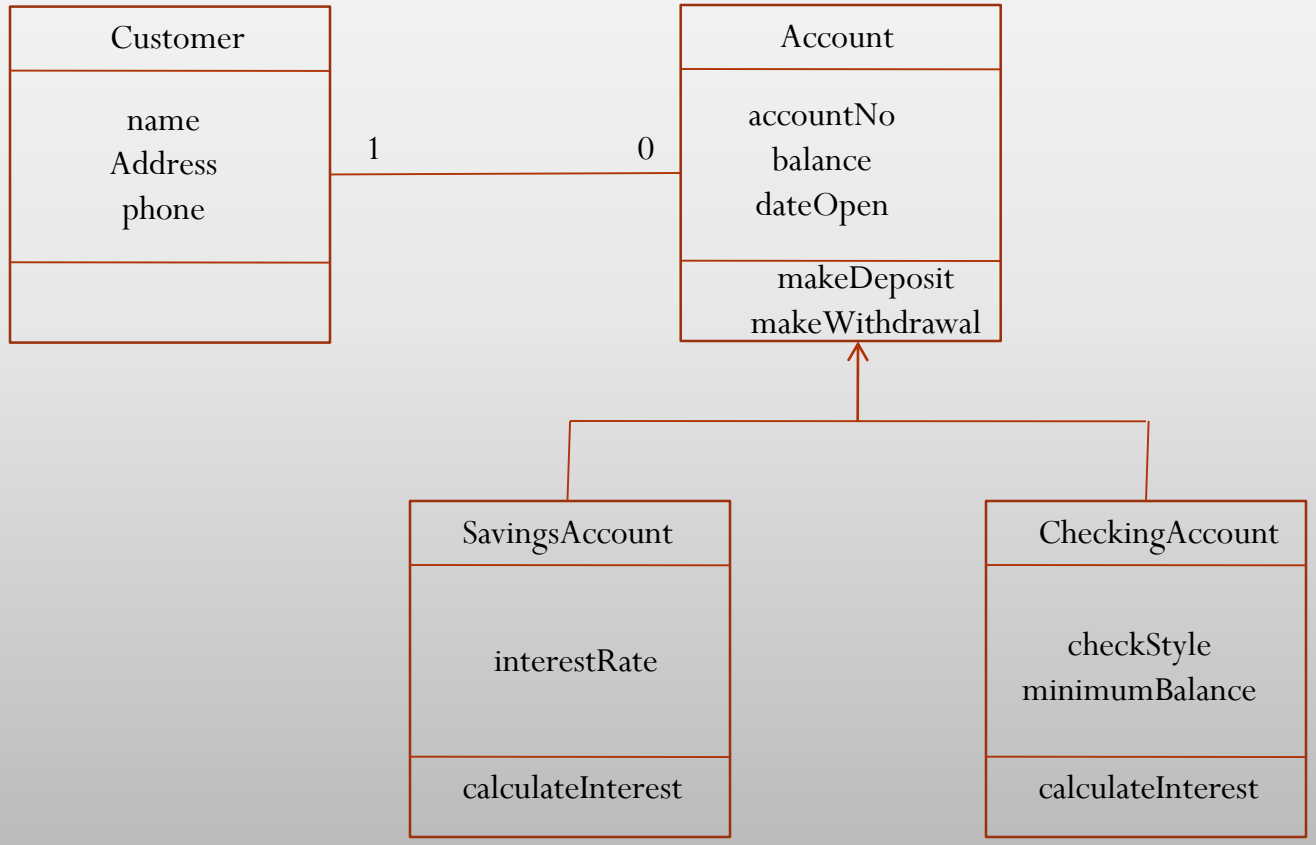
ຕ້ອງ ຈະດຳເນີ ນການໃຫ້

ຂໍ້ມູນລູກຄ້າທີ່ຊື່ ສົມ ສີ ໂຊກຊ່ວຍໄດ້ຖືກ ບັນທຶກລົງໃນໃບສັ່ງຊື້ ແລ້ວ

ອັດຖຸ Customer: ສົມສີ ໂຊກຊ່ວຍ
ລະຫັດລູກຄ້າ: 76532
ທີ່ຢູ່: ດົງໂດກ, ໄຊທານີ, ນະຄອນຫຼວງວຽງຈັນ

ຕ້ອງ ຈະດຳເນີ ນການໃຫ້

ຕົວຢ່າງ Class Diagram ທີ່ສ້າງຂຶ້ນລະຫວ່າງການວິເຄາະລະບົບແບບວັດຖຸ



ການພັດທະນາລະບົບແບບອັດຖ

ເຕັກນິກການພັດທະນາລະບົບແບບໂຄງສ້າງໄດ້ໃຊ້ແນວຄິດການແບ່ງສ່ວນ ຂໍ້ມູນ ແລະ ຂະບວນການອອກຈາກກັນ ເຊິ່ງໃນໂລກຂອງຄວາມເປັນຈິງສິ່ງເຫຼົ່ານີ້ຄວນຈະເກີດຂຶ້ນໃນເວລາດຽວກັນ ໂດຍເປັນໄປຕາມແນວຄິດແບບອັດຖ ເນື່ອງຈາກລະບົບຈະຖືກເບິ່ງເປັນອັດຖ ແລະ ໃນຕົວອັດຖກໍເປັນແຫ່ງລວມຂໍ້ມູນ ແລະ ວິທີການ ເຊິ່ງມີ Class ເປັນຕົວກຳນົດຄຸນສົມບັດໃຫ້ກັບອັດຖ.

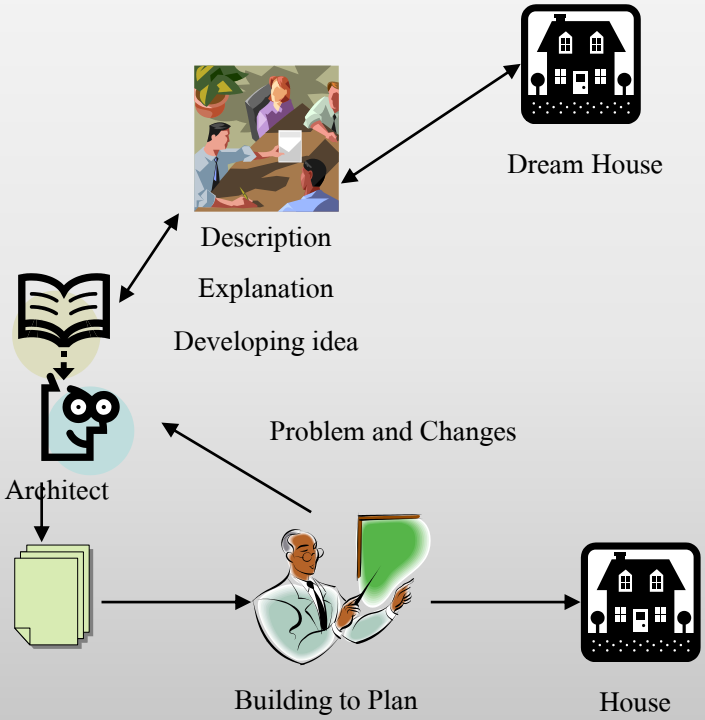
ປຽບທຽບການພັດທະນາລະບົບ

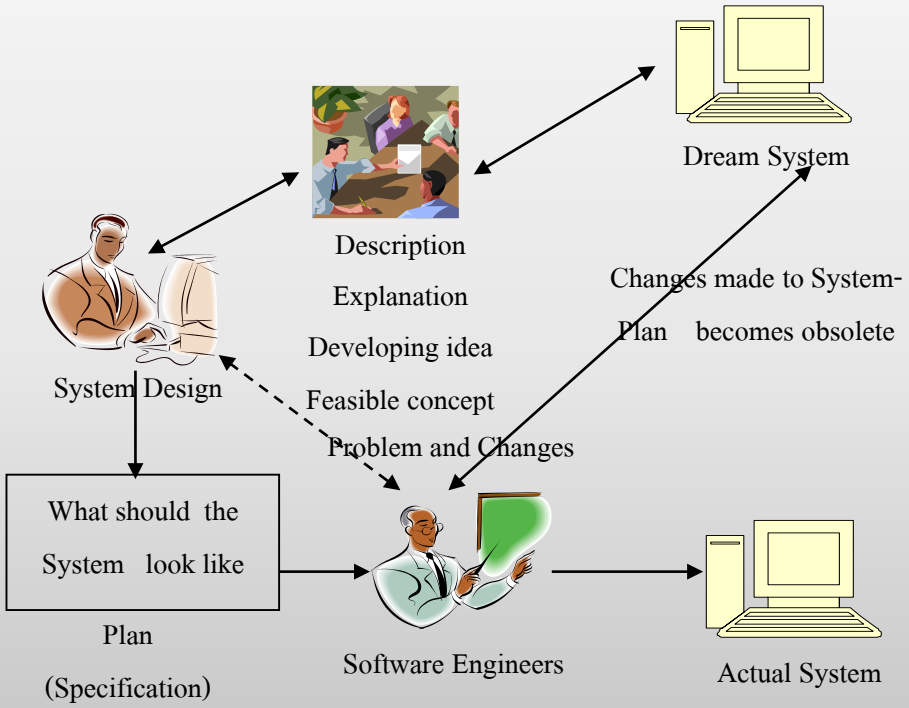
ແບບໂຄງສ້າ	ແບບອັດຖຸ
ວິເຄາະຂໍ້ມູນຈາກເອກະສານ, ລາຍງານ ແລະ ຂັ້ນຕອນການທຳງານຂອງລະບົບເດີມ	ວິເຄາະອັດຖຸທີ່ກ່ຽວຂ້ອງ
ແຕກຂະບວນການທຳງານອອກເປັນໂມດູນຍ່ອຍໆ	ຈັດກຸ່ມ, ຈັດປະເພດຂອງວັດຖຸທີ່ເປັນໄປຕາມຄຸນລັກສະນະ
ໂປຣແກຣມຕ່າງໆໃນລະບົບຈະມີຄວາມກ່ຽວຂ້ອງກັນ ເຊິ່ງການປ່ຽນແປງທີ່ເກີດຂຶ້ນອາດຈະສົ່ງຜົນກະທົບຕໍ່ໂປຣແກຣມອື່ນໆທີ່ກ່ຽວຂ້ອງ	ວັດຖຸແຕ່ລະຕົວຈະມີຄວາມເປັນເອກະລາດກັນ ດັ່ງນັ້ນ ການປ່ຽນແປງລາຍລະອຽດໃດໆ ຈະບໍ່ມີຜົນກະທົບເຊິ່ງກັນ ແລະ ກັນ
ການປັບປຸງ ຫຼື ປ່ຽນແປງລະບົບ ຈະເຮັດໄດ້ດ້ວຍການແກ້ໄຂ Source Code	ການປັບປຸງ ຫຼື ປ່ຽນແປງລະບົບ ຈະເຮັດໄດ້ດ້ວຍການປ່ຽນແປງລາຍລະອຽດຂອງຄຸນສົມບັດ ແລະ ຟັງຊັ່ນການທຳງານຂອງວັດຖຸນັ້ນໆ
ເຄື່ອງມືທີ່ນຳມາສະໜັບສະໜູນລະບົບປະຈຸບັນເລີ່ມມີໜ້ອຍລົງ	ເຄື່ອງມືທີ່ນຳມາສະໜັບສະໜູນລະບົບມີຫຼາຍຂຶ້ນ

ວິສະວະກຳຊອບແວ (Software Engineering)

ໃນຊຸມປີ ຄສ 1968 ໄດ້ມາການນຳເອົາຫຼັກການທີ່ເອີ້ນວ່າ ວິສະວະກຳຊອບແວ ເຂົ້າມາມີບົດບາດສຳຄັນຕໍ່ຂະບວນການພັດທະນາຊອບແວເພື່ອໃຫ້ການພັດທະນາຊອບແວມີມາດຕະຖານ ແລະ ເປັນວິທະຍາສາດຫຼາຍຂຶ້ນ ໂດຍຈະເຫັນໄດ້ວ່າຂະບວນການພັດທະນາຊອບແວນັ້ນເປັນສິ່ງທີ່ຈັບຕ້ອງຍາກ ດັ່ງນັ້ນ ຈຶ່ງມີຄວາມພະຍາຍາມນຳເອົາຫຼັກວິທະຍາສາດເຂົ້າມາປະຍຸກໃຊ້ເພື່ອໃຫ້ການພັດທະນາຊອບແວມີຄວາມແນ່ນອນ, ຊັດເຈນ, ມີຄວາມເປັນມາດຕະຖານ ແລະ ມີຄຸນນະພາບ.

ປຽບທຽບຂັ້ນການສ້າງເຮືອນ ແລະ ຂັ້ນຕອນການພັດທະນາລະບົບ





ຂະບວນການທາງວິສະວະກຳຊອບແວ

Sommerville ໄດ້ລະບຸກິດຈະກຳພື້ນຖານຂອງຂະ ບວນ ການທາງວິສະວະກຳຊອບແວໄວ້ຢູ່ 4 ສ່ວນຫຼັກໆຄື:

1. ຂໍ້ກຳນົດຊອບແວ (Software Specification)
2. ການພັດທະນາຊອບແວ (Software Development)
3. ການກວດສອບຄວາມຖືກຕ້ອງຂອງຊອບແວ (Software Validation)
4. ວິວັດທະນາການຂອງຊອບແວ (Software Evolution)

ຂໍ້ກຳນົດຊອບແວ (Software Specification)

ເປັນການລະບຸຂໍ້ກຳນົດໂດຍການກຳນົດຟັງຊັນໜ້າທີ່ຂອງຊອບແວ, ລວມເຖິງເງື່ອນໄຂຂໍ້ບັງຄັບການປະຕິບັດງານເທິງໜ້າທີ່ທີ່ຈະຕ້ອງໄດ້ຮັບການກຳນົດຂຶ້ນ, ກິດຈະກຳດັ່ງກ່າວເອີ້ນວ່າ ວິຊາວະກຳຄວາມຕ້ອງການ ເຊິ່ງເປັນຂັ້ນຕອນທີ່ສຳຄັນໃນຂະບວນການທີ່ປະກອບດ້ວຍ 4 ສ່ວນດັ່ງລຸ່ມນີ້:

- ສຶກສາຄວາມເປັນໄປໄດ້ (Feasibility study)
- ວິເຄາະຄວາມຕ້ອງການ (Requirements Analysis)
- ສະຫຼຸບເປັນຂໍ້ກຳນົດ (Requirement Specification)
- ກວດສອບຄວາມຕ້ອງການ (Requirements Validation)

ການພັດທະນາຊອບແວ (Software Development)

ແມ່ນການພັດທະນາ ຫຼື ສ້າງຜະລິດຕະພັນໃຫ້ກົງກັບຂໍ້ກຳນົດ ດ້ວຍການນຳເອົາແນວທາງວິທີການພັດທະນາຊອບແວມາໃຊ້ກັບການພັດທະນາຊອບແວເພື່ອໃຫ້ຂະບວນການພັດທະນາຊອບແວນັ້ນມີມາດຕະຖານ ແລະ ຕ້ອງຜະລິດຕະພັນຊອບແວມີຄຸນນະພາບ.

ການກວດສອບຄວາມຖືກຕ້ອງຂອງຊອບແວ

ຊອບແວຈະຕ້ອງໄດ້ຮັບການກວດສອບຄວາມຖືກຕ້ອງ ເພື່ອໃຫ້ແນ່ໃຈວ່າຊອບແວທີ່ໄດ້ພັດທະນາຂຶ້ນມານີ້ເປັນ ຜະລິດຕະພັນທີ່ກົງກັບຄວາມຕ້ອງການຂອງລູກຄ້າ ຫຼື ຜູ້ໃຊ້ ງານ.

ວິວັດທະນາການຂອງຊອບແວ (Software Evolution)

ການພັດທະນາຊອບແວຂະໜາດໃຫຍ່ມີຄວາມຊັບຊ້ອນຫຼາຍ ແລະ ມັກຈະມີໄລຍະເວລາໃນການພັດທະນາທີ່ຍາວນານ ດັ່ງນັ້ນ ລະບົບທີ່ພັດທະນາຈະມີວິວັດທະນາການກວດສອບຂໍ້ຜິດພາດຕາມຂໍ້ກຳນົດເດີມ, ແຕ່ກໍອາດຈະມີຂໍ້ກຳນົດໃໝ່ໆເຂົ້າມາ ສະນັ້ນ ຊອບແວອາດຈະມີການປ່ຽນແປງໄປຕາມຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ລະຫວ່າງການພັດທະນາ ແລະ ກໍອາດຈະເປັນໄປໄດ້ວ່າລະບົບຍ່ອຍຕ່າງໆເຊິ່ງບາງລະບົບຍັງບໍ່ມີຄວາມເປັນເອກະລາດໃນຕົວເອງ ດັ່ງນັ້ນ ໃນລະບົບຍ່ອຍໃດໜຶ່ງກໍຈະສົ່ງຜົນສະທ້ອນຕໍ່ລະບົບຍ່ອຍອື່ນໆ ສະນັ້ນ ຊອບແວຄວນອອກແບບໄວ້ເພື່ອຮອງຮັບວິວັດທະນາການໃນການປ່ຽນແປງດັ່ງກ່າວ ເພື່ອໃຫ້ເປັນໄປຕາມຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ງານໄດ້ຢ່າງເໝາະສົມ

ຄຸນລັກສະນະຂອງ Software ທີ່ມີຄຸນນະພາບ

- ມີຄວາມຖືກຕ້ອງ (Correctness) .
- ມີຄວາມເຊື່ອຖືໄດ້ (Reliability).
- ໃຊ້ງ່າຍ (User friendliness.
- ງ່າຍຕໍ່ການປ່ຽນແປງ (Adaptability).
- ສາມາດນຳເອົາມາໃຊ້ງ່າຍໄດ້ອີກ (Reusability).
- ສາມາດເຂົ້າກັບລະບົບຕ່າງໆໄດ້ (Interoperability).
- ມີປະສິດທິພາບ (Efficiency).
- ມີຄວາມສະດວກໃນການເຄື່ອນຍ້າຍ (Portability).
- ມີຄວາມປອດໄພ (Security.

Models ການພັດທະນາຊອບແວ

ໂມເດິນການພັດທະນາຊອບແວ ແມ່ນແບບຈຳລອງທີ່ໃຊ້ສຳລັບເປັນຕົວຊີ້ນຳກິດຈະກຳຫຼັກໃນຂະບວນການພັດທະນຊອບແວ ດ້ວຍການກຳນົດລາຍລະອຽດຕ່າງໆໄວ້ໃນແຕ່ລະກິດຈະກຳຕາມແຕ່ລະຂັ້ນຕອນທີ່ມີລຳດັບໄວ້ຢ່າງຊັດເຈນ ເພື່ອຕ້ອງການໃຫ້ຂະບວນການພັດທະນາຊອບແວດຳເນີນຕໍ່ໄປມີບັນຫາໜ້ອຍທີ່ສຸດ ແລະ ສາມາດນຳມາປະຍຸກໃຊ້ກັບການພັດທະນາຊອບແວຕັ້ງແຕ່ເລີ່ມຕົ້ນຈົນຈົບ ໂດຍໃຫ້ຜູ້ໃຊ້ໄດ້ເຫັນພາບລວມ ແລະ ເຂົ້າໃຈຂັ້ນຕອນຕ່າງໆຂອງລະບົບໄດ້ທັນທີ

Models ການພັດທະນາຊອບແວ

1. Built and Fix Model
2. Water Fall Model
3. Incremental Model
4. Spiral Model
5. Rapid Application Development (RAD)
6. Joint Application Development (JAD)
7. Ration Unified Process (RUP)

Built and Fix Model

ເປັນ Model ທີ່ພັດທະນາຊອບແວ ດ້ວຍການຂຽນໂປຣແກຣມ ແລະ ແກ້ໄຂໄປເລື້ອຍໆ ດ້ວຍການລອງຜິດລອງຖືກ ຈົນພໍໃຈ ຫຼື ຄິດວ່າຜົນທີ່ໄດ້ຮັບກົງກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ແລ້ວ ເຊິ່ງຂະບວນການດັ່ງກ່າວຈະເຮັດໃຫ້ເສຍເວລາໄປກັບການ Debug ໂປຣແກຣມ ແລະ ບໍລຸງຮັກສາລະບົບ ແຕ່ຢ່າງໃດກໍຕາມການພັດທະນາຊອບແວດ້ວຍວິທີນີ້ ອາດຈະໃຊ້ງານໄດ້ດີກັບໂປຣແກຣຂະໜາດນ້ອຍທີ່ບໍ່ມີຄວາມຊັບຊ້ອນຫຼາຍ ຫຼື ເໝາະກັບງານເມື່ອມີການເກີດຂໍ້ຜິດພາດຈະບໍ່ສົ່ງຜົນກະທົບຕໍ່ລະບົບຫຼາຍ, ແຕ່ບໍ່ເໝາະກັບການພັດທະຍາຊອບແວຂະໜາດໃຫຍ່ ເນື່ອງຈາກການພັດທະນາຊອບແວຂະໜາດໃຫຍ່ຈຳເປັນຕ້ອງມີການວາງແຜນ, ການວິເຄາະ ແລະ ການອອກແບບ.

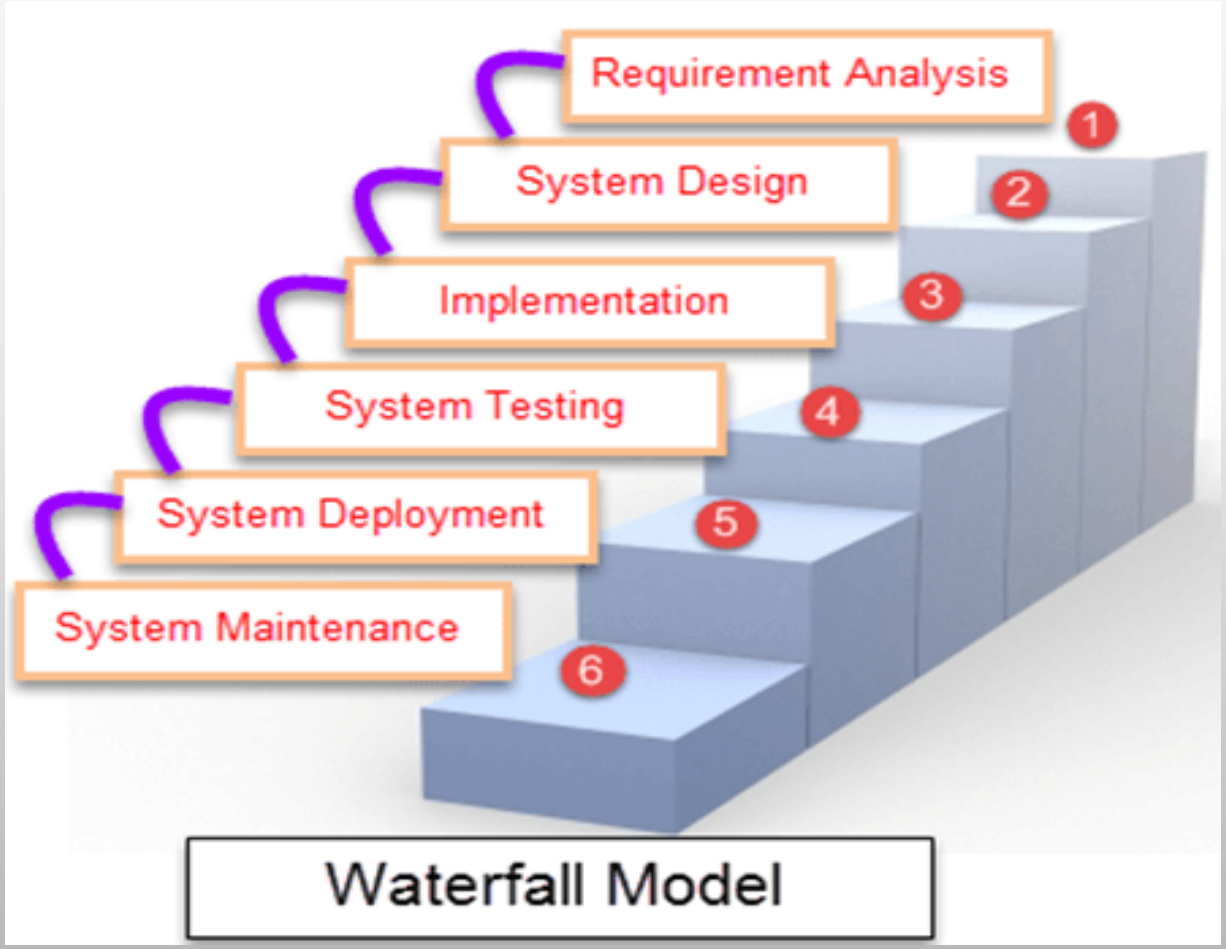
ຂັ້ນຕອນຂອງໂມເດິນ Built and Fix Model

1. ຂຽນໂຄດຄຳສັ່ງໂປຣແກຣມບາງສ່ວນທີ່ຄາດວ່າຈະສາມາດແກ້ໄຂບັນຫາ.
2. ຄອມພາຍ ແລະ ຣັນໂປຣແກຣມເພື່ອທົດສອບ
3. ຖ້າພົບຂໍ້ຜິດພາດໃນໂປຣແກຣມກໍດຳເນີນການແກ້ໄຂ
4. ກັບໄປເຮັດຊ້ຳຕາມຂັ້ນຕອນ 1-4 ຈົນໄດ້ຕາມຄວາມຕ້ອງການ

Water Fall Model

Water Fall Model ເປັນແບບທີ່ມີມາດົນນານແລ້ວຕັ້ງແຕ່ ປີ ຄສ 1970 ແລະ ເປັນທີ່ນິຍົມໃຊ້ໃນການພັດທະນາລະບົບຈົນເຖິງປະຈຸບັນ, ເນື່ອງຈາກໂມເດິນນີ້ງ່າຍຕໍ່ການນຳມາ ປະຍຸກໃຊ້ງານ. ເຊິ່ງໂມເດິນການພັດທະນາແບບນີ້ມີຄວາມຄ້າຍຄືກັບວົງຈອນການພັດທະນາລະບົບຕາມແນວທາງຂອງ SDLC(System Development Life Cycle) ໂດຍຈະເຫັນ ໄດ້ວ່າຂະບວນການພັດທະນາລະບົບແບບນ້ຳຕົກດັ້ງເດີມນັ້ນເມື່ອມີການເຂົ້າສູ່ຂັ້ນຕອນໃດໆແລ້ວຈະບໍ່ມີການຍ້ອນກັບມາເຮັດຂັ້ນຕອນກ່ອນໜ້ານັ້ນໄດ້ອີກ, ແຕ່ໃນຄວາມເປັນຈິງແລ້ວຂັ້ນຕອນການ ຍ້ອນກັບໄປເຮັດຂັ້ນຕອນກ່ອນໜ້ານັ້ນສາມາດເກີດຂຶ້ນໄດ້ ເຊັ່ນວ່າ ນັກວິເຄາະອາດຈະເຫັນບັນ ຫາ ຫຼື ຂໍ້ຜິດພາດທີ່ເກີດຂຶ້ນກ່ອນໜ້ານັ້ນ ຫຼື ມີບາງຢ່າງທີ່ຈຳເປັນຕ້ອງໄດ້ຮັບການປ່ຽນ ແປງຈຶ່ງ ຕ້ອງກັບໄປແກ້ໄຂບັນຫາດັ່ງກ່າວ, ເຊິ່ງການກະທຳດັ່ງກ່າວເປັນການຢືນຢັນເຖິງຄວາມຕ້ອງການ ໃນລະບົບນັ້ນໃຫ້ກົງກັບຄວາມຕ້ອງການ ຂອງຜູ້ໃຊ້ຈິງ. ດັ່ງນັ້ນ ຈຶ່ງໄດ້ມີການປັບປຸງ Model ນີ້ໃໝ່ດ້ວຍການເພີ່ມຄຸນສົມບັດຂອງການວົນຊ້ຳໃຫ້ເປັນວົງຮອບທັງນີ້ກໍເພື່ອໃຫ້ສາມາດກັບຄືນ ໄປແກ້ໄຂບັນຫາ ຫຼື ຂໍ້ຜິດ ພາດໃນຂັ້ນຕອນກ່ອນໜ້ານັ້ນໄດ້.

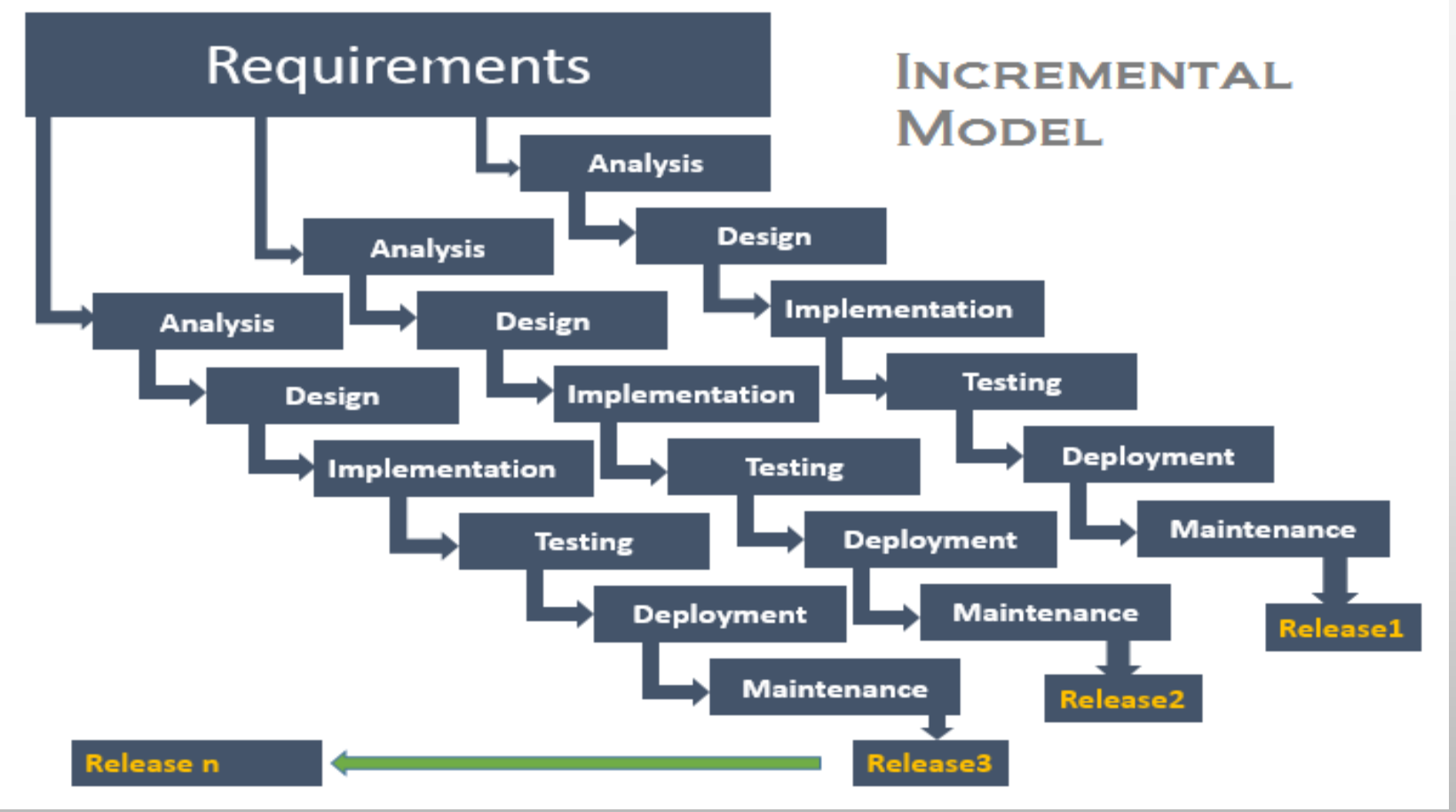
Water Fall Model



Incremental Model

Incremental Model ເປັນ Model ໜຶ່ງທີ່ໄດ້ນຳເອົາຫຼັກການຂອງ Water Fall Model ມາທຳການປັບປຸງປະສິດທິພາບໃຫ້ດີຍິ່ງຂຶ້ນ. ດັ່ງທີ່ຮູ້ກັນແລ້ວວ່າ Water Fall Model ຈະມີຂໍ້ບົກ ຜ່ອງຢູ່ບ່ອນວ່າມີຂະບວນການທົດສອບຢູ່ຕອນທ້າຍໆ ເຊິ່ງມີໂອກາດຈະຍ້ອນກັບໄປຈຸດເລີ່ມຕົ້ນຂອງ ໂຄງການໃໝ່ທັງໝົດຖ້າມີການວາງແຜນ ຈັດການບໍ່ດີພໍ. ດັ່ງນັ້ນ Incremental Model ຈຶ່ງສາມາດ ຫຼຸດຂໍ້ບົກຜ່ອງຂອງ Water Fall Model ໄດ້ດ້ວຍການເພີ່ມສ່ວນຂອງ ການອອກແບບ ແລະ ພັດທະນາໃນຮູບແບບ ຂອງສ່ວນງານຍ່ອຍໆໃນລັກສະນະແບບກ້າວໜ້າ (increment) ໂດຍແຕ່ລະສ່ວນ ງານ ຍ່ອຍຈະມີການວົນຊ້ຳເປັນຮອບໃນລັກສະນະ Interaction ພ້ອມກັບມີລະບົບການກວດ ສອບ.

Incremental Model



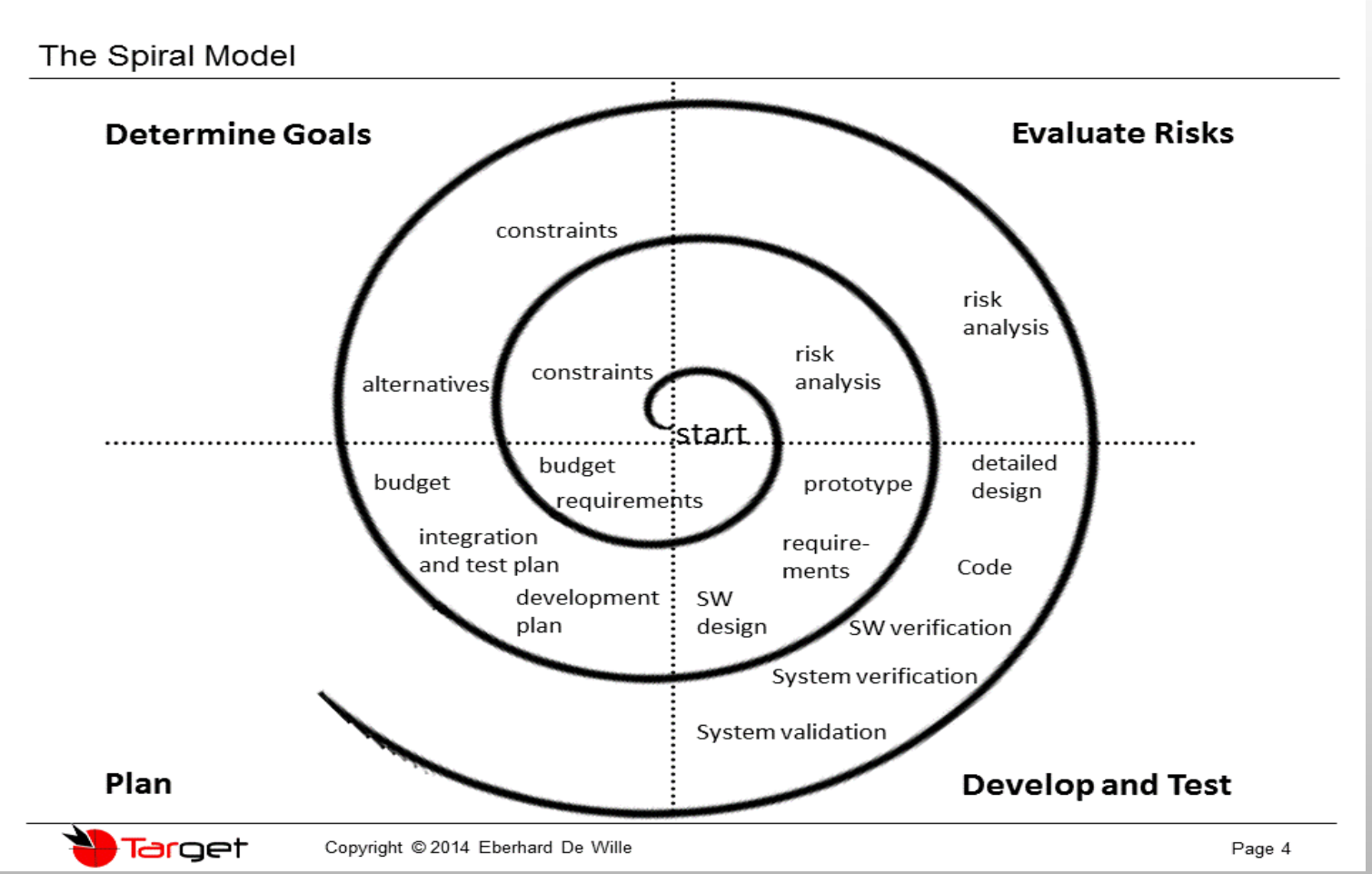
Spiral Model

Spiral Model ເປັນ Model ໜຶ່ງທີ່ມີຫຼັກການທຳງານເປັນລັກສະນະວົງມົນວົນເປັນກົ້ນຫອຍເຊິ່ງຈະວົນໄປຕາມທິດຂອງເຂັມໂມງດ້ວຍການວົນໄປເລື້ອຍໆ, ໂດຍແຕ່ລະວົງຮອບນັ້ນຈະປະກອບດ້ວຍຂັ້ນຕອນທີ່ສຳຄັນດັ່ງນີ້

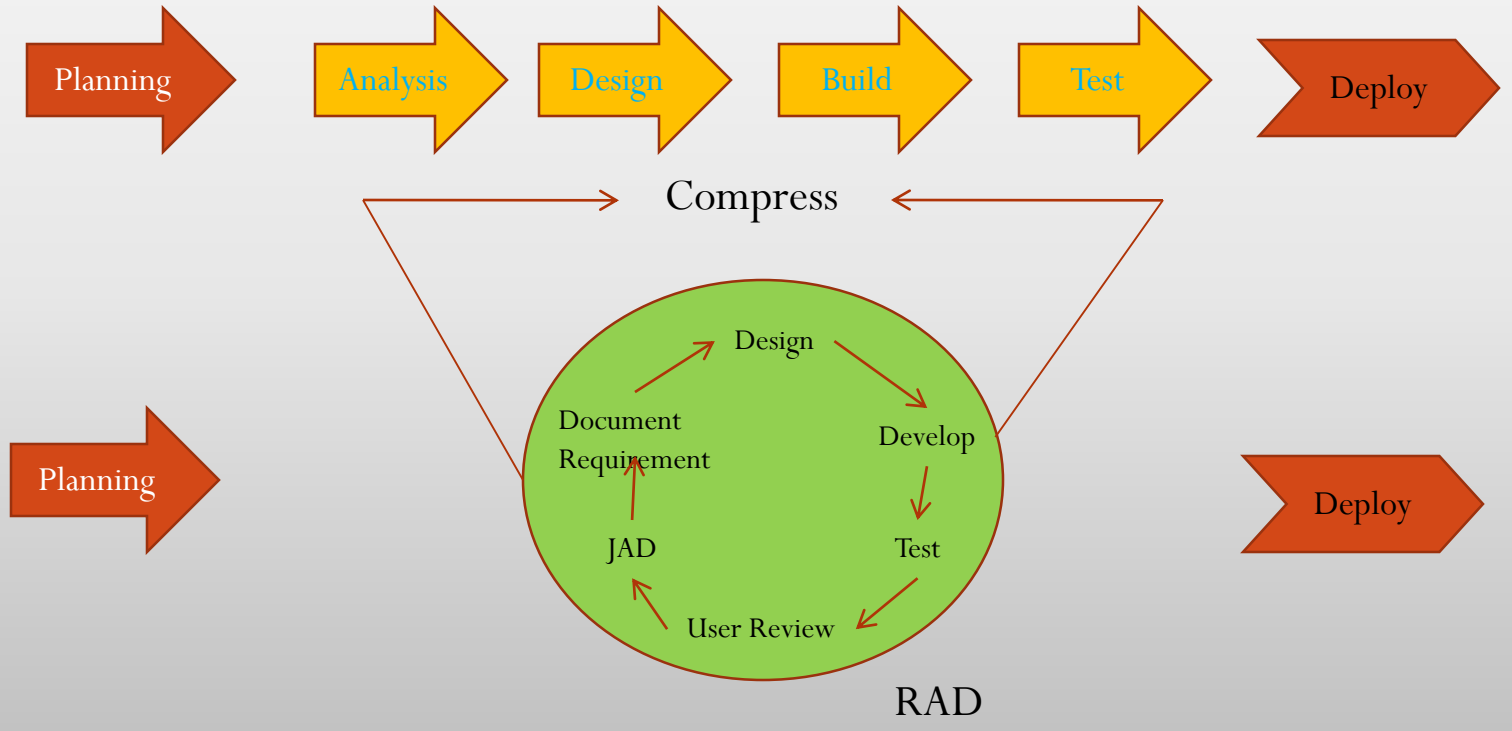
- ການວິເຄາະຫາຄວາມຕ້ອງການ (Requirement Analysis)
- ການວິເຄາະຫາຄວາມສ່ຽງ (Risk Analysis)
- ການອອກແບບຕົ້ນແບບ (Design Prototype)
- ການພັດທະນາຕົ້ນແບບ ແລະ ການນຳມາປະກອບກັນ (Develop and Integrate Prototype)

ເມື່ອໄດ້ປະຕິບັດຕາມຂັ້ນຕອນນີ້ແລ້ວກໍຈະໄດ້ຕົ້ນແບບທີ່ສົມບູນໃນແຕ່ລະວົງຮອບ, ຈາກນັ້ນກໍຈະດຳເນີນການເຊັ່ນເດີມໃນຮອບຕໍ່ໄປ. ເຊິ່ງແບບຈຳລອງນີ້ຈະໃຊ້ໄດ້ຢ່າງເໝາະສົມ ກັບລະບົບງານທີ່ມີ ໂອກາດປ່ຽນແປງເລື້ອຍໆ, ເນື່ອງຈາກໃນແຕ່ລະຮອບນັ້ນຈະມີການວິເຄາະຫາຄວາມຕ້ອງການໃໝ່ ແລະ ວິເຄາະຫາຄວາມສ່ຽງວ່າຈະທຳການພັດທະນາຕໍ່ໄປ ຫຼື ບໍ່ ຫຼືພຽງພໍແລ້ວສຳລັບຮອບນີ້ເທົ່ານັ້ນ, ໂດຍຄວາມສ່ຽງທີ່ນຳມາວິເຄາະນັ້ນຈະພິຈາລະນາໃນດ້ານຄວາມສ່ຽງຂອງໂຄງການ (Project Risk), ຄວາມສ່ຽງຂອງຜະລິດຕະພັນ (Product Risk) ແລະ ຄວາມສ່ຽງທາງທຸລະກິດ (Business Risk)

Spiral Model



Rapid Application Development (RAD)



RAD

ຈຸດປະສົງຂອງ RAD ແມ່ນຕ້ອງການລວບລວມຂະບວນການສຳຄັນຕ່າງໆເພື່ອ ພັດທະນາລະບົບດ້ວຍໄລຍະເວລາທີ່ສັ້ນ ໂດຍໃຊ້ CASE Tools ເຂົ້າມາຊ່ວຍ

Joint Application Development (JAD)

JAD ແມ່ນເຕັກນິກການພັດທະນາຮ່ວມກັນ ເຊິ່ງປະກອບດ້ວຍບຸກຄົນທີ່ມີສ່ວນຮ່ວມໃນອົງກອນ ແລະ ຜູ້ຊ່ຽວຊານທາງດ້ານ IT ດ້ວຍການເຂົ້າຮ່ວມປະຊຸມທາງປະຕິບັດການ (Workshop) ຢ່າງມີເປົ້າໝາຍ. ເຊິ່ງຈຸດປະສົງຫຼັກຂອງ JAD ແມ່ນການພັດທະນາລະບົບໂດຍໃຊ້ເວລາສັ້ນ ແລະ ຜົນໄດ້ຮັບໂຄງການມີຄວາມສົມບູນ. ເປົ້າໝາຍ Workshop ຂອງ JAD ກໍຄື:

1. ສົນທະນາກັນເພື່ອແລກປ່ຽນແນວຄວາມຄິດແລະ ເປັນເອກະພາບກັນ
2. ມີກຸ່ມຜູ້ໃຊ້ເຂົ້າຮ່ວມປະມານ 3-5 ຄົນ ແລະ ໃຫ້ມີນັກພັດທະນາທີ່ມີຄວາມຊ່ຽວຊານດ້ານ IT ປະມານ 2-3 ຄົນ
3. ມີຜູ້ທີ່ມີອຳນາດທີ່ສາມາດຕັດສິນໃຈໃນກໍລະນີມີຄວາມຂັດແຍ່ງກັນ
4. ມີຜູ້ສັງເກດການປະມານ 2-3 ຄົນ ທີ່ເຮັດໜ້າທີ່ສັງເກດການເທົ່ານັ້ນ
5. ຕ້ອງມີຜູ້ທີ່ຊ່ຽວຊານໃນລະບົບທຸລະກິດ

Ration Unified Process (RUP)

RUP ເປັນການພັດທະນາລະບົບແບບວັດຖຸທີ່ຖືກພັດທະນາຂຶ້ນ ໂດຍ Grady Booch, James Rumbaugh ແລະ Ivar Jacobson ເຊິ່ງໃຊ້ Rational Software. ໂດຍທັງ 3 ຄົນນີ້ເປັນຜູ້ຢູ່ເບື້ອງຫຼັງຂອງພາສາ UML (Unified Modeling Language)

ຈຸດປະສົງຂອງ RUP ແມ່ນຕ້ອງການໃຫ້ທີມງານພັດທະນາຊອບແວມີຄຸນນະພາບສູງກົງກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ພາຍໃຕ້ງົບປະມານ ແລະ ເວລາທີ່ກຳນົດໄວ້. ໂດຍພື້ນຖານຂອງຂະບວນການນີ້ກໍຄືການສ້າງ Model ແລະ ການຈັດ Model ດ້ວຍພາສາ UML.

ເຄື່ອງມືທີ່ໃຊ້ສະໜັບສະໜູນການພັດທະນາລະບົບ (Tools to Support System Development)

- ເຄື່ອງມືສ້າງແຜນວາດ (Diagramming Tools)
- ເຄື່ອງມືຈັດທຳຄຳອະທິບາຍ (Description Tools)
- ເຄື່ອງມືສ້າງຕົ້ນແບບ (Prototyping Tools)
- ເຄື່ອງມືຈັດການດ້ານຄຸນນະພາບ (Quality Management Tools)
- ເຄື່ອງມືຈັດທຳເອກະສານ (Documentation Tools)
- ເຄື່ອງມືການອອກແບບ ແລະ ແປງລະຫັດ (Design and Code Generator Tools)

ມາດຕະຖານຂອງລະບົບຂ່າວສານ

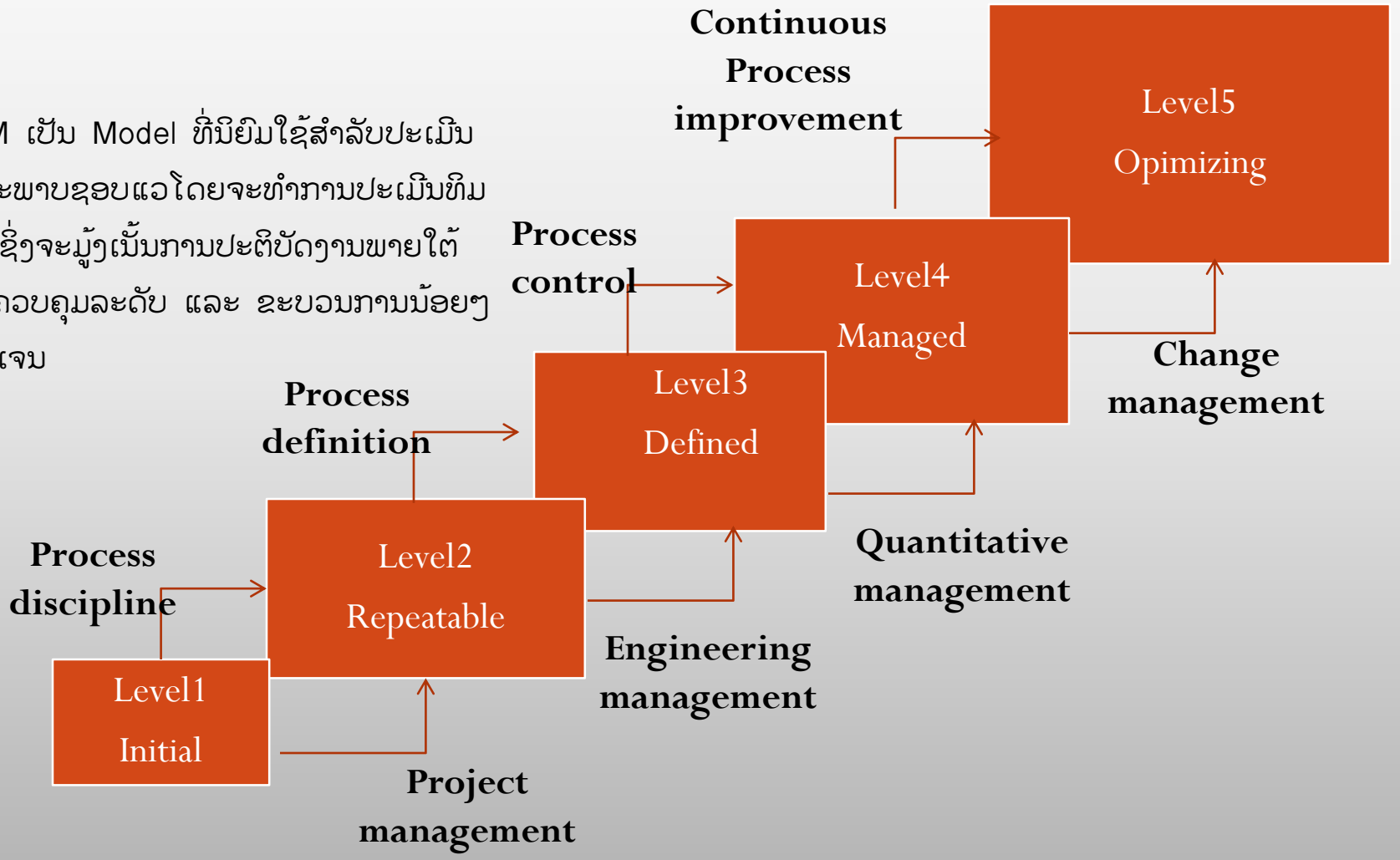
ມາດຕະຖານທີ່ໃຊ້ກັບຜະລິດຕະພັນຊອບແວທີ່ນິຍົມ ແລະ ຍອມຮັບກັນທົ່ວໂລກກໍຄື Capability Maturity Model: CMM ທີ່ພັດທະນາຂຶ້ນໂດຍສະຖາບັນວິສະວະກຳຊອບແວ (Software Engineering Institute: SEI). ເນື່ອງຈາກວິກິດການທີ່ຊອບແວສ່ວນໃຫຍ່ມັກມີຄຸນນະພາບຕ່ຳ ແລະ ບໍ່ກົງກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ລວມທັງເວລາ ແລະ ຄ່າໃຊ້ຈ່າຍທີ່ປະມານໄວ້ບໍ່ກົງກັບແຜນທີ່ວາງໄວ້ ເນື່ອງຈາກຫຼາຍບັນຫາດັ່ງນີ້:

ບັນຫາໃນໂຄງການພັດທະນາຊອບແວ

- ການສື່ສານບໍ່ສັດເຈນລະຫວ່າງຜູ້ພັດທະນາ ແລະ ຜູ້ໃຊ້ ເຊິ່ງເຮັດໃຫ້ຜູ້ພັດທະນາພັດທະນາຊອບແວບໍ່ກົງກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້
- ບໍ່ສາມາດຈັດການກັບຄວາມສ່ຽງລວມທັງການປ່ຽນແປງຄວາມຕ້ອງການໃໝ່ໆທີ່ອາດຈະເກີດຂຶ້ນໃນອານາຄົດໃນລະຫວ່າງການພັດທະນາ
- ສະມະຊິກໃນທີມງານມີຂະບວນການເຮັດວຽກເປັນຂອງຕົນເອງເຊິ່ງບໍ່ມີມາດຕະຖານທີ່ແນ່ນອນເຮັດໃຫ້ຍາກຕໍ່ການຕິດຕາມ
- ບັນຫາການລວມໂປຣແກຣມຈາກທີມງານເນື່ອງຈາກທີມງານບໍ່ມີມາດຕະຖານດຽວກັນ
- ຊອບແວບໍ່ມີຄຸນນະພາບເນື່ອງຈຳເປັນຕ້ອງໄດ້ບຳລຸງຮັກສາເປັນປະຈຳ
- ຊອບແວມີຄວາມຊັບຊ້ອນຫຼາຍເກີນໄປ
- ຄວາມບໍ່ເອົາໃຈໃສ່ຂອງຫົວໜ້າໂຄງການຕໍ່ການບໍລິຫານທີມງານຫຼືບໍ່ມີປະສົບປະການ
- ຊອບແວບໍ່ມີຄວາມໜ້າເຊື່ອຖື, ບໍ່ມີລະບົບການກວດສອບທີ່ຊັດເຈນ ແລະ ຂາດເຄື່ອງມື

Model CMM

CMM ເປັນ Model ທີ່ນິຍົມໃຊ້ສຳລັບປະເມີນ ຄຸນນະພາບຊອບແວໂດຍຈະທຳການປະເມີນທີມ ງານເຊິ່ງຈະມຸ້ງເນັ້ນການປະຕິບັດງານພາຍໃຕ້ ການຄວບຄຸມລະດັບ ແລະ ຂະບວນການນ້ອຍໆ ທີ່ຊັດເຈນ



Thank you

Q and A