Designing a Measurement Method for the Portability Non-Functional Requirement (NFR)

Feras AbuTalib

Electrical Engineering Department
McGill University
Montreal, Québec,Canada

Alain Abran

École de technologie supérieure
Université du Québec
Montreal, Québec, Canada

Dennis Giannacopoulos

Electrical Engineering Department
McGill University
Montreal, Québec, Canada

IWSM-MENSURA 2013

Overview

- Definitions of the **portability** NFR in the literature.
- Benefits of measuring portability
- Previous attempts to measure portability
- Steps required to design the new portability NFR measurement method
- Illustrative application
- Conclusion.

Definitions of Portability in the Literature

- Ease with which a system or component can be transferred from one hardware or software environment to another. (ISO/IEEE 24765)
- How well the software can adapt to changes in its environment or with its requirements. (ISO 9126)
- Capability of a program to be executed on various types of data processing systems without converting the program to a different language & with little or no modification. (ISO 2382-1)

Benefits of Measuring Portability

In software, the portability requirement has an impact on at least 2 parameters:

I. The technology used to develop the system.

- Determining the environments where the software system is expected to operate is a key point on choosing the technologies to develop the system.

II. The extensibility of the system.

- By correctly modeling the portability requirement, it becomes much easier to evaluate the possibilities/cost associated to use the software systems on new environments.

Previous Attempts to Measure Portability (1/2)

Very few attempts to measure the portability NFR.

- Mooney: not based on international standards & no clear distinction between the measurement method itself & the efforts/cost estimation model linked to the portability NFR.
- A standards-based measurement model: in "Computer Standards and Interfaces Journal" in 2013: a system view of the portability functions allocated to software.

Previous Attempts to Measure the Portability NFR (2/2)

- Our new measurement method differs from in many aspects including the methodology from which the measurement method was designed & the view point of the measurement.
  - This new measurement method should be more suitable for use by the software development & QA teams.

Steps Required to Design the New Portability NFR Measurement Method.

Step 1:

- Determination of the Measurement Objectives.

Step 2:

- Characterization of the concept to be measured.

Step 3:

- Design or Selection of the Meta Data.

Step 4:

- Numerical Assignment Rules

Step 1: Determination of the Measurement Objectives (1/2)

- **Objective:** to measure the functional size of the **portability** requirement.

- Measurement **point of view:** from the software development perspective.

Step 1: Determination of the Measurement Objectives (2/2)

- Intended use of the measurement result: to determine the functional size of the **portability** requirements for a software product **throughout** the software life cycle:
  - whether it has yet to be built or it has already been delivered.

- The results obtained from such measurements can be utilized in many ways such as estimating the needed development time and/or the quality assurance efforts to certify the software.

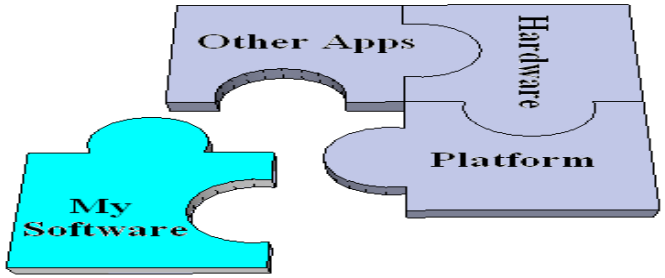
Step 2: Characterization of the Concept to be Measured (1/7)

- The portability requirement depends on the environments where the software should be operating on.
- A software product may interact in its environment with 3 main different components:
  I. **The Hardware.**
  II. **The Platform.**
  III. **Other Software Applications.**

Step 2: Characterization of the Concept to be Measured (2/7)

- **How to identify the components?**

  Think about your software system as a plug-in piece that is to be added to an existing environment.



Alternatively, think about the environment as a set of layers and the software application as the last layer.

It is important to identify only those distinct components that the software interacts directly with.

Step 2: Characterization of the Concept to be Measured (3/7)

- An environment can be modeled as an n-tuple:
  - The n elements of a single tuple are the components that the software system is interacting with.

- Example: the environment of a software browser add-in can be modeled with a 3-tuple (Intel Core i5, Microsoft Windows 7, Microsoft Internet Explorer browser 10).

Step 2: Characterization of the Concept to be Measured (4/7)

- **Not all 3 main components should always be in the model.**
  - Identify only the components that the software system would directly interact with.
    - Ex: Assembly programming.
    - Ex: MFC programs on MS Windows operating systems.

Step 2: Characterization of the Concept to be Measured (5/7)

- **In the model, you may have more than one component from the same category.**
  - Ex: Games Applications interacting with hardware (CPU + Graphic Cards)
  - Ex: A software application that depends on the existence of more than one other software applications.

Step 2: Characterization of the Concept to be Measured (6/7)

- **Different versions of the same component can be modeled as different components.**
  - This should be considered if the functionality of the software system is considerably different when interacting with different versions of the same components.
    - Ex: An add-in tool that can be deployed on Microsoft Internet Explorer Browser (Version 10, Version 9, Version 8, Version 7).
    - Ex: Backward compatibility on a series of a single platform (JVM 6 vs JVM 7)

Step 2: Characterization of the Concept to be Measured (7/7)

- **The entity to be measured :** The key is to identify the number of distinct environments' components where the software system should/will directly operate on or interact with.
- **Characterization of the concept to be measured**

  The software system talks with its environments components by means of interactions.

  - These interactions can be best modeled with the concept of COSMIC data movement to indicate a move (transfer) of one or more data attributes belonging to one system to another.

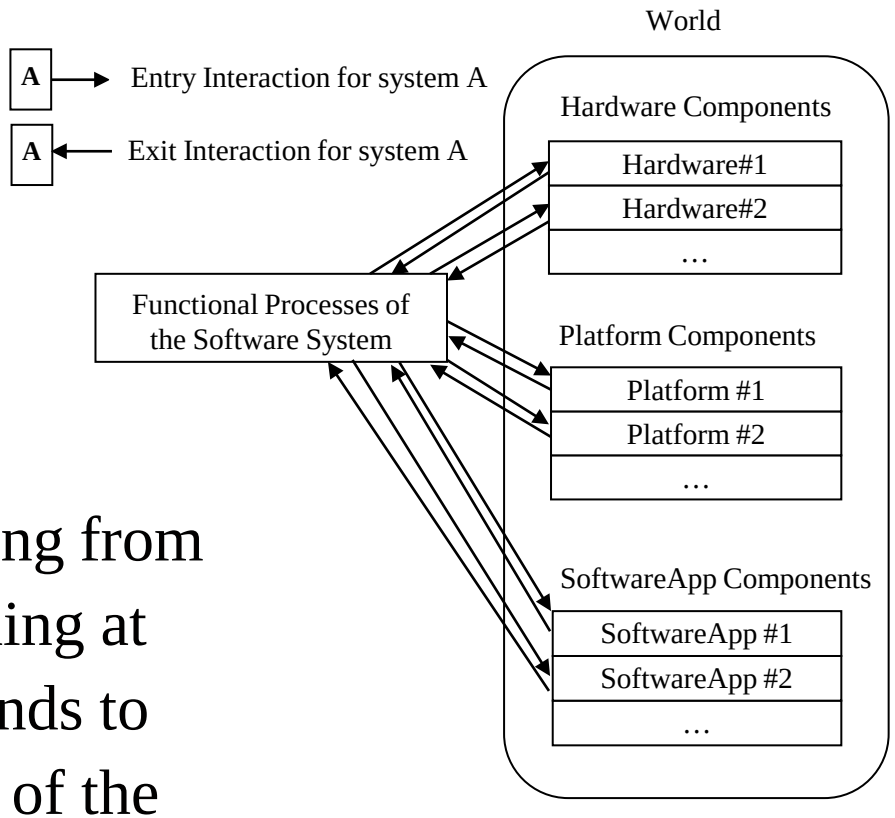
  - 2 possible interaction types: Entry and Exit interactions.

Step3: Designing of the Meta Model (1/2)

- The software system will be modeled into COSMIC functional process.
  - The functional process is defined as an elementary component of a set of functional user requirements, comprising a unique, cohesive and independently executable set of data movements. (ISO 19761)
- Each functional process may interact with the components of its environments independently from the other processes.

Step3: Designing of the Meta Model (2/2)

- The figure illustrates the Meta Model proposed.
  - The number of arrows originating from the functional processes corresponds to the number of Entry interactions.
  - The number of arrows originating from the other environments and ending at the functional process corresponds to the number of eXit interactions of the software system.



Step4: Numerical Assignment Rules (1/2)

Functional size of the portability requirement (FSP)

= number of the system interactions of the software system (its functional processes) and the components of its supported environments

= Total Number of Entry System Interactions + Total Number of Exit System Interactions

= Total Number of hardware System Interactions + Total Number of platform System Interactions + Total Number of application System Interactions

$$= \sum SystemInteractionsOfOnefunctionalProcess$$

Step4: Numerical Assignment Rules (2/2)

Measurement Scale Type

- Each system interaction is considered to be of the same size independently of the actual components or its type (Exit/Entry).
- A system interaction is assigned a numerical size of **1 Interaction Unit (IU)** based on the underlying COSMIC concepts of only moving one data group in each interaction.
- It is expected that any software in the real world to have at least one Entry or Exit interaction.
- **IU (Interaction Unit =1)** has a ratio scale type which means it can be used in statistical analysis with the admissible mathematical calculations.

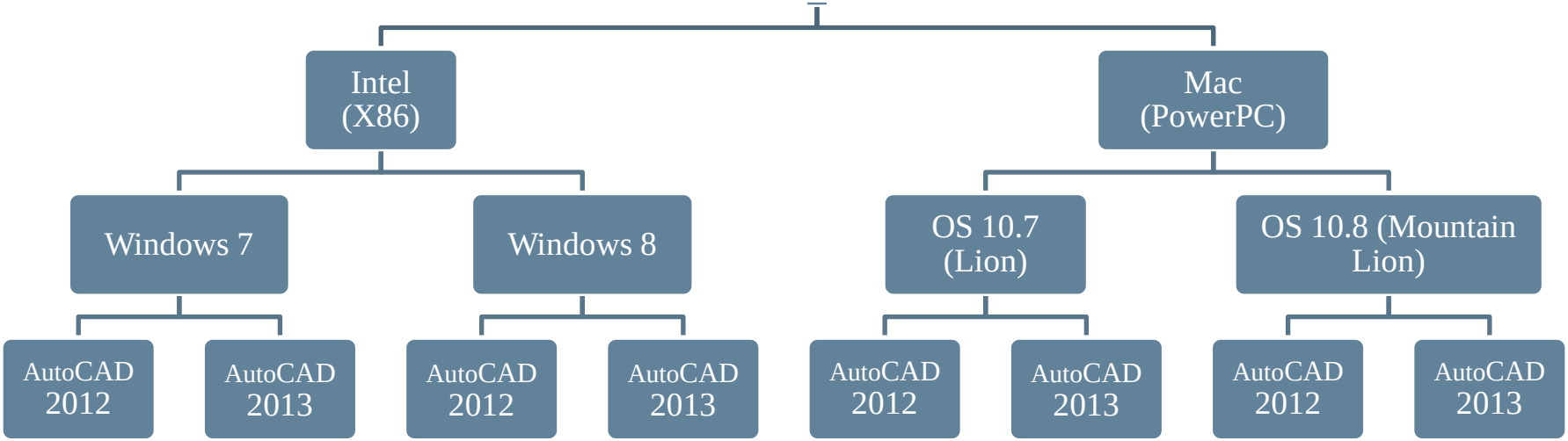
Illustrative Application (1/5)

- Developing an add-in tool on top of AutoCAD
  - One of the functional processes of this add-in is as follows: upon the startup of the AutoCAD application, the add-in will start a timer to obtain the time spent by the user in the application. The time will be written to a file on the user's machine when the session ends.
  - In the documentation, the portability requirement for this add-in is stated as follows: the add-in application should work on **version 2012** and **version 2013** of the **AutoCAD** application on the **Windows 7**, **Windows 8**, **Mac Lion**, and **Mac Mountain Lion** Operating Systems.

Illustrative Application (2/5)

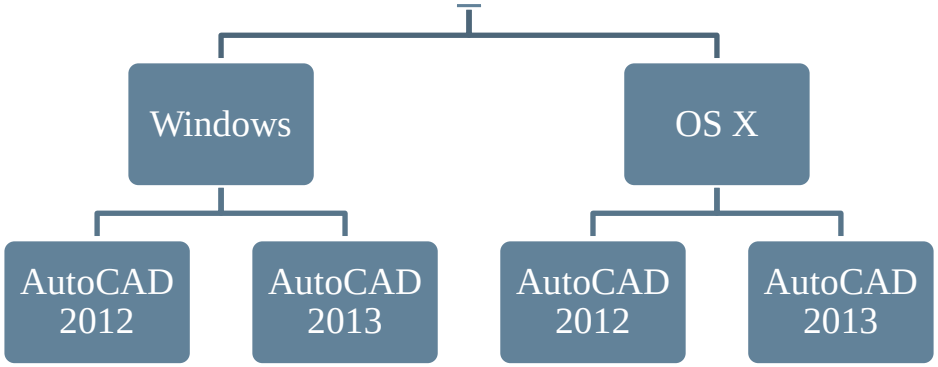
- The measurer should first identify the components of the environments that the add-in will be directly interacting with.

- One can identity the environments' components by visualizing the environments with a tree



Illustrative Application (3/5)

- The add-in software will not be interacting directly with any hardware components.
- The interaction done between the add-in and the windows platforms is the same regardless of its version (7 or 8).
- The same assumption applies also for the Mac OS X platforms.



Number of distinctive environments

= number of leaves in the tree

= 4

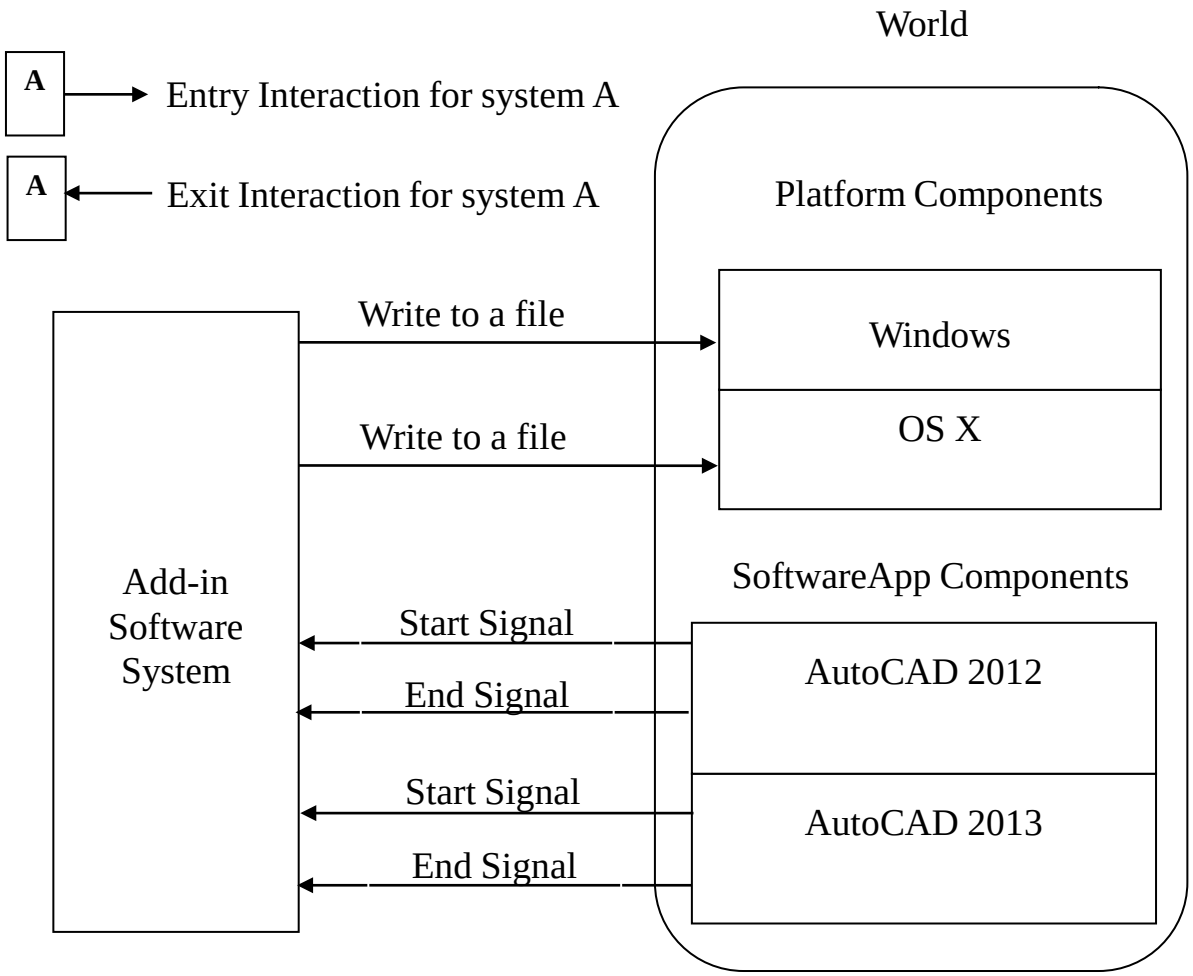
Illustrative Application (4/5)

- Even though there is no generic numerical formula rule which can be used to find the number of distinctive environments, applicants can still generate one based on their own data.

- Example: the following equation illustrates the formula rule that can be used when the developed software system should support the same number of platforms on each hardware components & the same number of software applications on each platform :

$$NumberOfDistictiveEnvironments = \sum HardwareComponents * \sum PlatformComponents * \sum ApplicationComponents$$

Illustrative Application (5/5)



- The number of interactions needed by the add-in software to accomplish the task of the time tracking functional process is six interactions.

Associated Possible Estimation Models (1/8)

- The value obtained from the proposed measurement method presented can be used directly in establishing estimation models.

- This can help for example the managers to better predict and evaluate the cost/effort needed to implement the development projects.

Associated Possible Estimation Models (2/8)

- Example: the manager of the development team of the previous add-in would like to estimate the cost associated to develop the functionalities of the add-in on different new possible environments.

- The manager can derive a simple candidate estimation model that is based on the measured value obtained from the proposed measurement method.

Associated Possible Estimation Models (3/8)

1) the manager would need to find out the $Portability\_coefficient_{New/Ref}$ ratio.
   - This is a unit-less ratio with the following formula:

     $$Portability\_coefficient_{New/Ref} = (FSP_{New} - FSP_{Shared}) / FSP_{Ref}$$

     Where FSP_Z is the functional size of the portability NFR of the software under the Z environment.
   - The ratio should be larger than or equal to zero.
   - If the ratio is calculated to be negative then the value of zero must be used instead.

Associated Possible Estimation Models (4/8)

2) the portability coefficient$_{New/Ref}$ ratio can be used in the following formula to carry on the process to establish the estimation model:

$$Effort_{New} = portability\_coefficient_{New/Ref} * Effort_{Ref} + E$$

Where:

- ***Effort$_Z$*** *is the total effort needed in working hours units to develop the software functionalities on Z environment.*
- ***E*** *is the error/extra term value in working hours units which represents the contribution of all other variables not included in the main independent variable explicitly stated.*

Associated Possible Estimation Models (5/8)

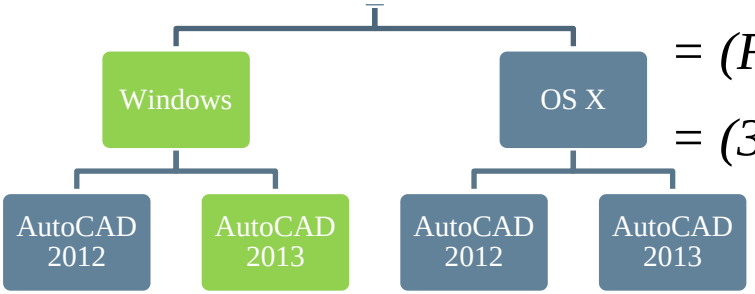
- Assume that the add-in development team started by developing the add-in for 1 possible environment, a single branch of the tree, such as the (Windows, AutoCAD2012).
- Assume it took 12 working hours to accomplish this task.
- The effort to develop the functionality of the add-in for the (Windows, AutoCad 2013) environment can be estimated to be :

$$= portability\_coeff_{New/Ref} * Effort_{Ref} + E$$

$$= portability\_coeff_{(Win,Cad2013)/(Win,Cad2012)} * Effort_{(Win,Cad2012)} + E$$

$$= (FSP_{(Win,Cad2013)} - FSP_{Shared(win)}) / FSP_{(Win,Cad2012)} * 12 + E$$

$$= (3\text{-}1)/3 * 12 + E \quad = 2/3 * 12 + 0 = 8 \text{ working hours}$$



Associated Possible Estimation Models (6/8)

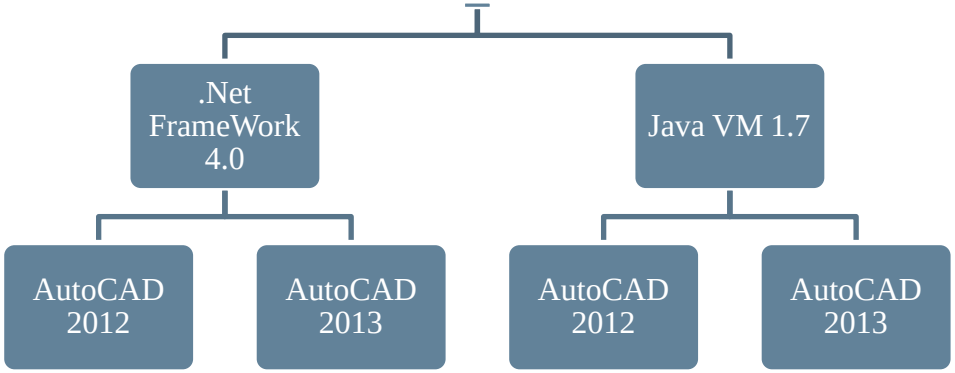
- The E error/extra term in the estimation formula can be used to consider other factors, such as the extra effort that is needed to support some specific differences between two Operating Systems from the same platform series.

  (ex: Windows 7 VS Windows 8 for the Windows Platform series).

- This E term can be modeled as well or obtained from previous similar projects completed by the development team and stored in their history data repository.

Associated Possible Estimation Models (7/8)

- Consider the following modification to the add-in application.
  Instead of relying directly on the OS platform components, it was decided to rely on the .Net Framework 4.0 on Windows environments and the Java Virtual Machine (JVM) 1.7 on Mac environments.



Associated Possible Estimation Models (8/8)

- Both the functional size of the portability NFR for the .Net Framework and the Java VM is equal to 1 interaction unit.

- But in terms of efforts, one language might be easier/faster to develop with.

- In practice, this may depends on many factors such as: staff experience, available reusable code, number of developers, etc. If such information is provided, then it can be modeled in the E term to come up with a better estimation model.

Conclusion

Points to remember:

- The proposed measurement method of the Portability requirement was taken from the development point of view.

- The proposed measurement method of the Portability requirement depends heavily on correctly identifying the environments where the software product should be operating on and/or interacting with.

- In the same development process, it is normal that different teams model the same portability requirement differently based on their point of view

Conclusion

- The proposed measurement method can be used to obtain a base measure for the functional size of the portability non functional requirement of the software being developed and/or examined.

- This base measure can be used later with or in derived formulas in the analysis models to obtain indicators.

- These indicators can be interpreted to produce the information product that would typically be used in the evaluation process or the decision making process.

Questions???

? ?

?