Automated Software Sizing Based on Sequence and Class Diagrams

Kan Qi, Barry Boehm

{kqi, boehm}@usc.edu

Outline

- Motivations for automated software sizing
- Existing Methods
- Functional Software Sizing Models
- UML Metamodels
- Bridge the gap: User-system Interaction Model (USIM)
- The Automated Counting Process of DUCPs
- Evaluation
- Conclusions

Motivations for automated software sizing

Project Effort/software size estimation at the early stages of a project is essential for various software management decisions (most of effort estimation models require size metric as the major predictor).

- Control the project scope to avoid risks.
- Make project plans.
- Resource allocation.

Software Size Estimation is costly and inaccurate.

- Intensively involves manual effort.
- Requires expertise to produce reproducible results.

Existing Methods

Methods to automate the existing size metrics:

- Fetcke proposed an approach to map use case model, domain object model, analysis model to FPA's abstraction of a software system.
- Umeura proposed the rules to count Function Points from sequence and class diagrams.

Size metrics based on UML diagrams:

- Predictive Object Points (POP)
- Class Points
- UML Points
- The Fast&Serious Method

Our method formalizes the mapping from UML metamodels to the functional sizing model. Based on the formalization, an effective size metric called Detailed Use Case Points (DUCPs) is proposed and validated.

Functional Software Sizing Models

Use Cases

Use cases model the interactions between actors and a system. Use Cases are weighted by the number of transactions. The sum of the weighted use cases is used to estimate the software size. This model is used by Use Case Points (UCPs).

$$Software\ size = \sum_{c \in C} w_c$$

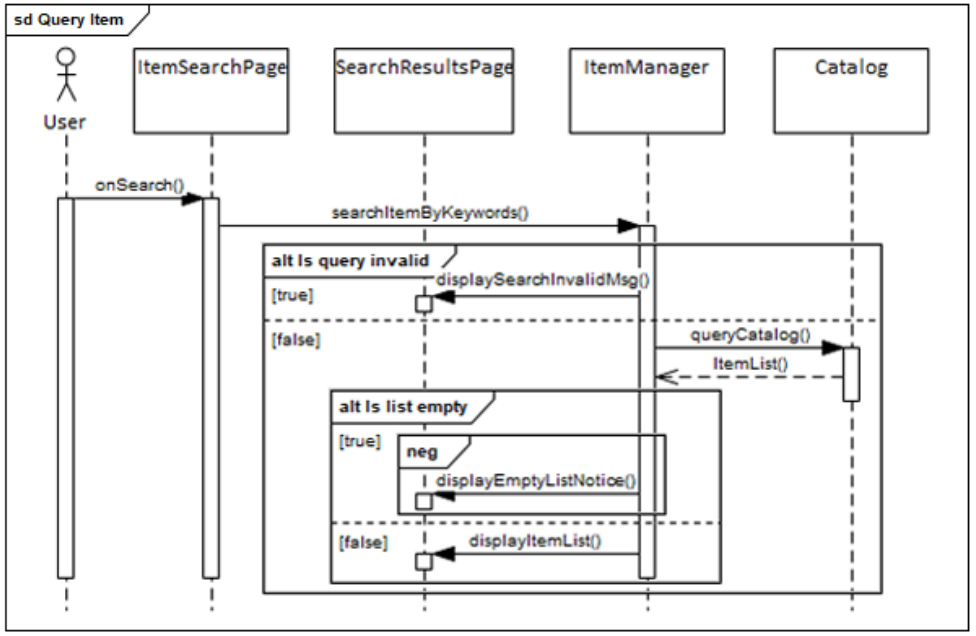
Transactions

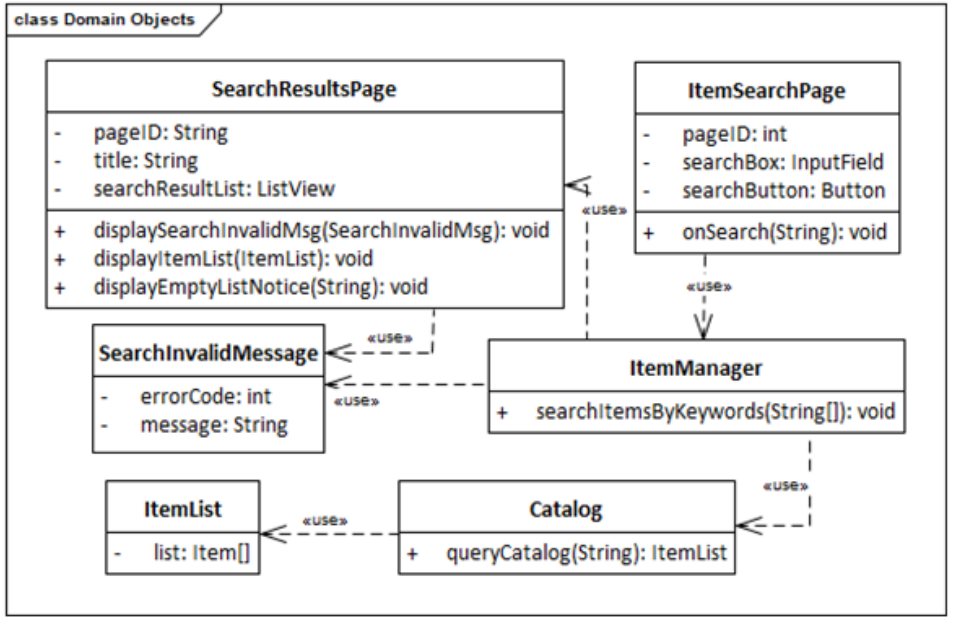
A transaction is a sequence of interactions between system components, which realizes a basic unit of system functionality. Transactions are weighted by their internal structures (data elements, components, activities, etc) to represent its contribution to the overall effort. This model is used by Function Points.

$$Software\ Size = \sum_{t \in T} w_t$$

UML Diagrams

- Sequence diagrams model the behavioral aspect of a system by sequences of messages.
- Class diagrams model the structural aspect of a system by classes in terms of attributes and operations.
- In the proposed automated software sizing method, we use the information derived from this two types of diagrams.





UML Metamodel

- Sequence and class diagrams themselves can be formally defined by class diagrams, which is called UML metamodel.

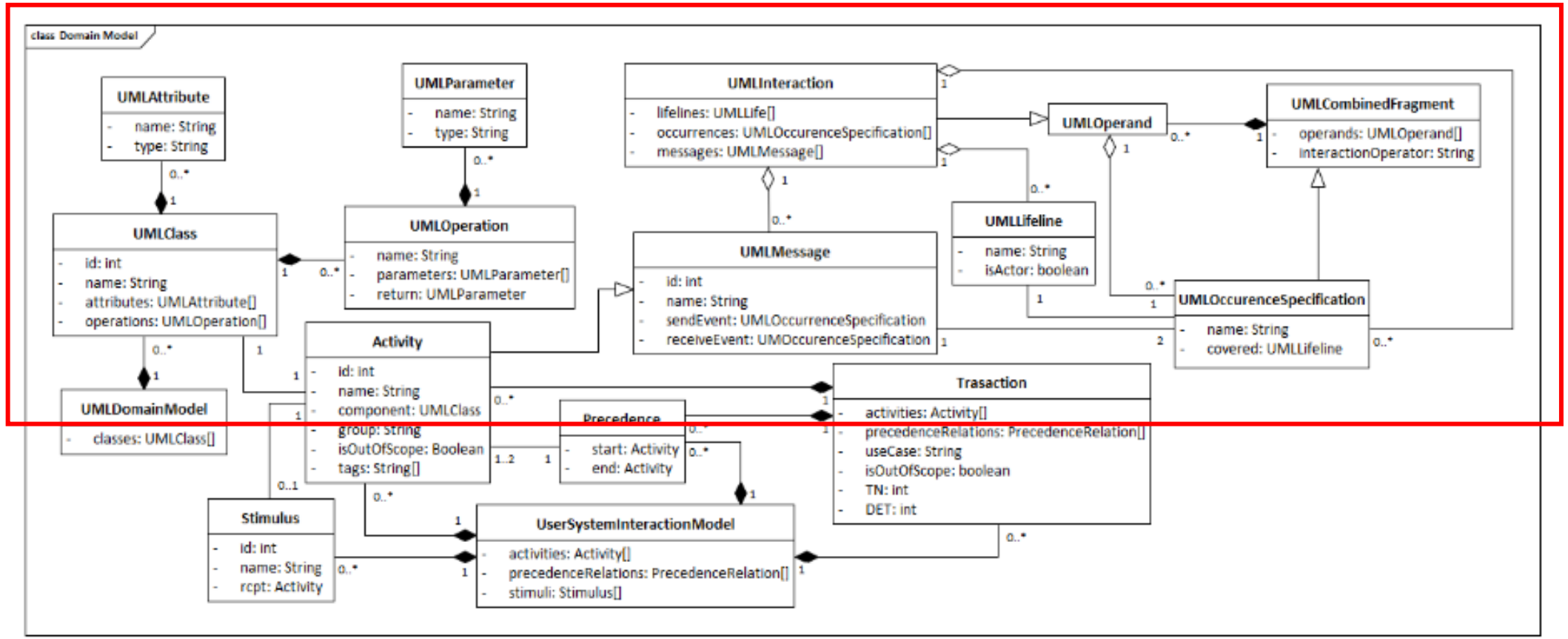
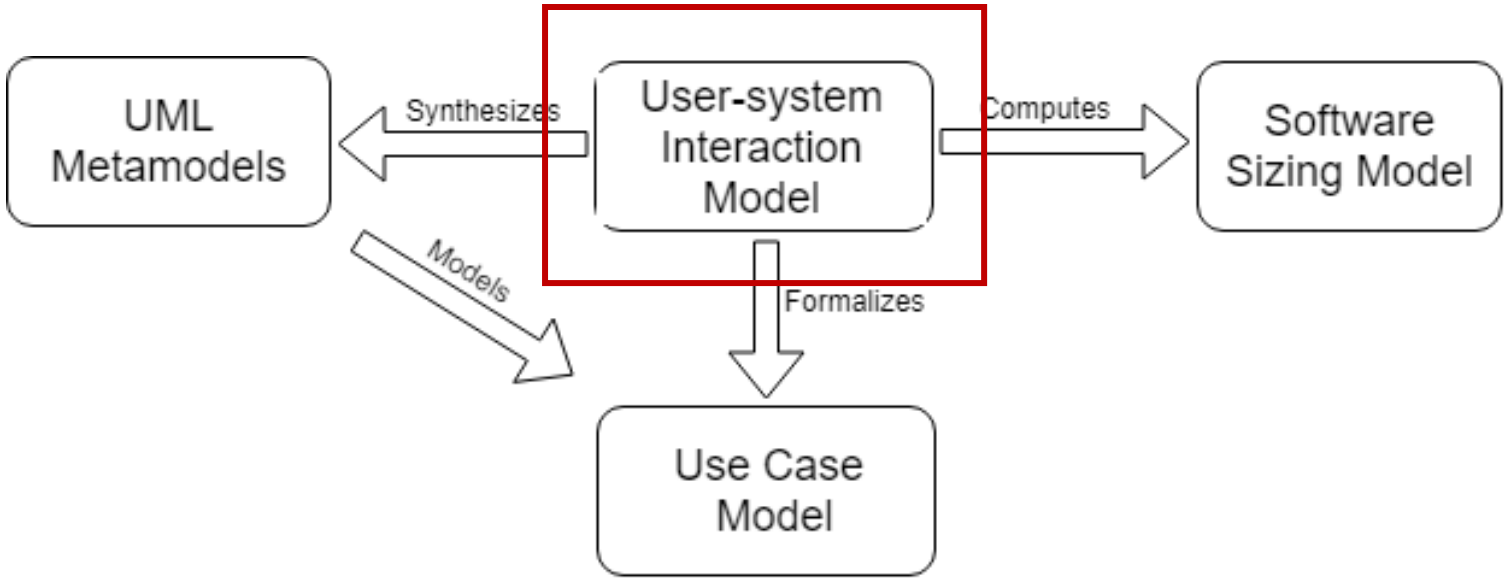


Figure 5: Domain Elements for User-system Interaction Model (USIM)

Bridge the gap between Software Sizing Model and UML Metamodels



Properties of the proposed user-system Interaction Model:

- Abstracts/synthesizes the metamodels of sequence and class diagrams.
- Computable for mathematical software sizing models.
- Provides implementation perspective for Use Case Model.

User-system Interaction Model

User-system interaction mode is defined as a tuple <A, PR, SB, SCP, STL, CMP> including 6 types of key elements:

- Activities (A)
- Precedence Relations (PR)
- System Boundary (SB)
- System Scope (SCP)
- Stimuli (STL)
- Components (CMP)

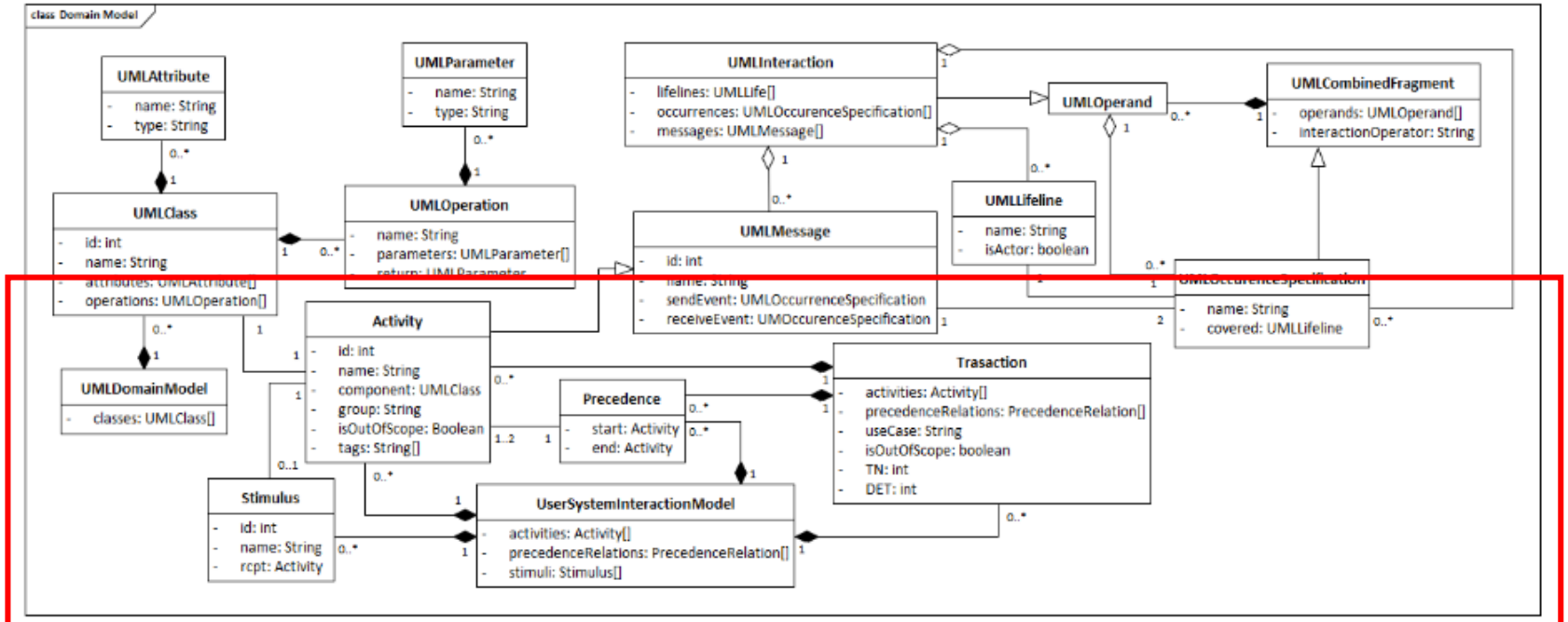
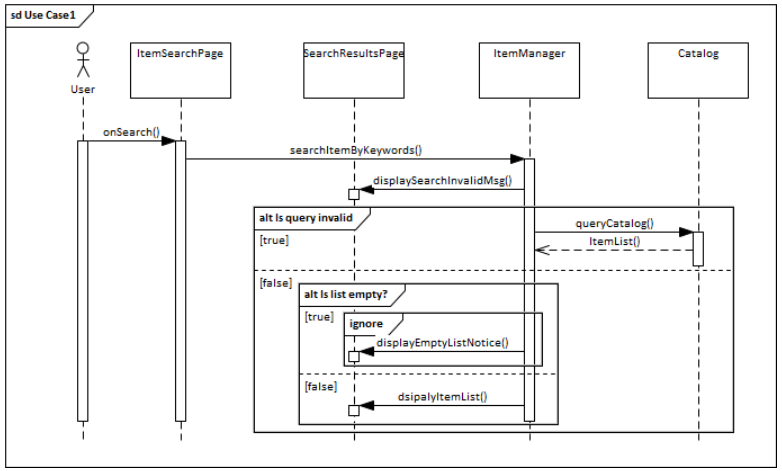


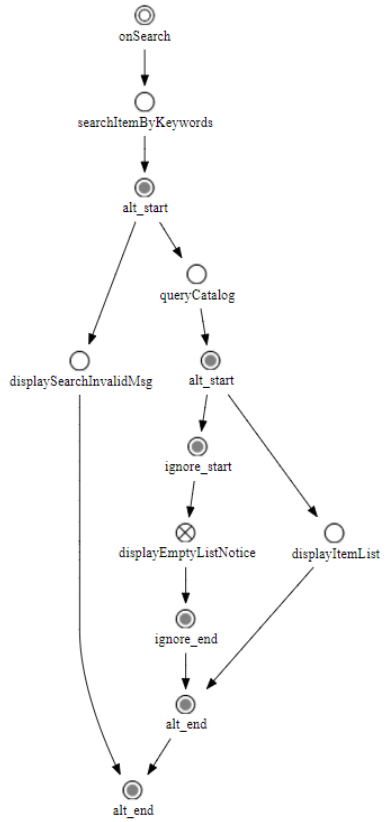
Figure 5: Domain Elements for User-system Interaction Model (USIM)

Model Construction

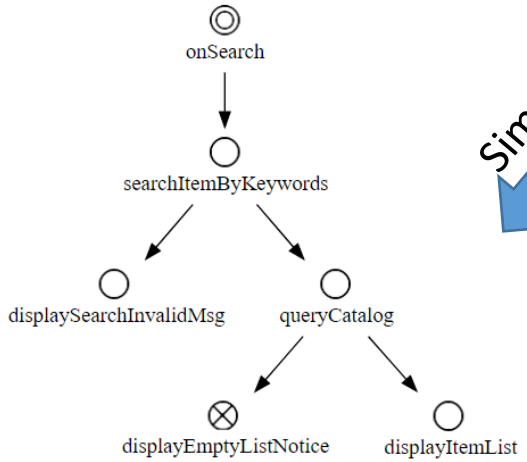
Sequence Diagrams are transformed into SDCFG <A, PR>:



Transformation



Simplification



Model Construction

- Components (CMP):

$$cmp(\alpha) = \{c | m.receiveEvent = o \,\&\, o.covered = l \,\&\, l.sName = c.sName \,\&\, msg(\alpha) = m \,\&\, o \in O \,\&\, l \in L \,\&\, c \in C\}$$

- Stimuli (STL):

$$RSP = \{\alpha | m.sendEvent = o \,\&\, o.covered = l \,\&\, l.isActor = true \,\&\, msg(a) = m \,\&\, o \in O \,\&\, l \in L\}$$

$$STML = \{\alpha' | \alpha'.rcpt = \alpha \,\&\, \alpha' id = m.id \,\&\, \alpha \in RSP \,\&\, m \in M\}$$

- System Boundary (SB):

$$grp(\alpha) = \{l.name | m.receiveEvent = o \,\&\, o.covered = l \,\&\, l.isActor = true \,\&\, msg(\alpha) = m \,\&\, o \in O \,\&\, l \in L\}$$

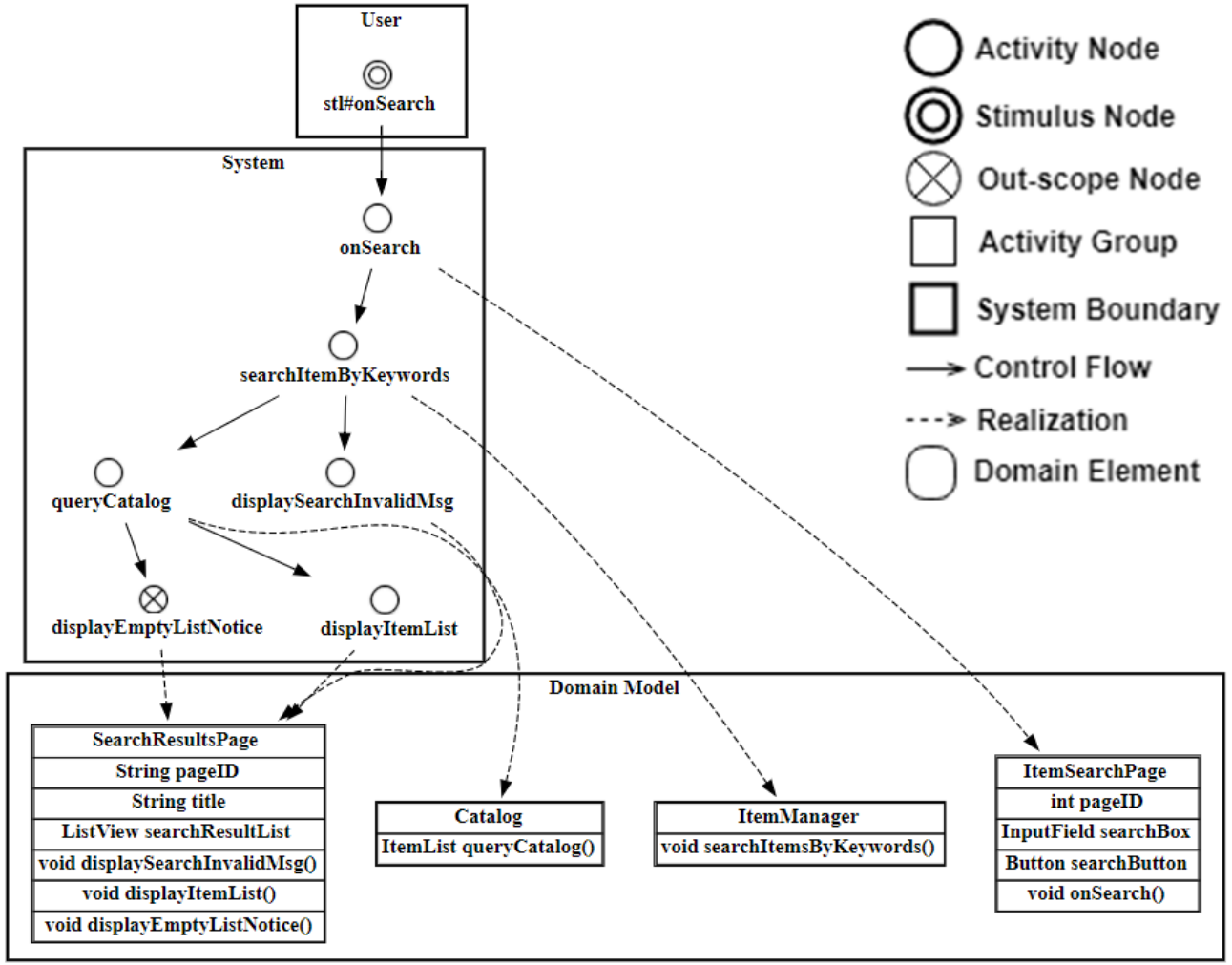
$$A_{system} = A - \cup_{g \in G}(\alpha \in g)$$

- System Scope (SCP):

$$isOutOfScope(\alpha) = (\text{"option"} \in \alpha.tags \,||\, \text{"negative"} \in \alpha.tags)$$

User-system Interaction Model Visualization

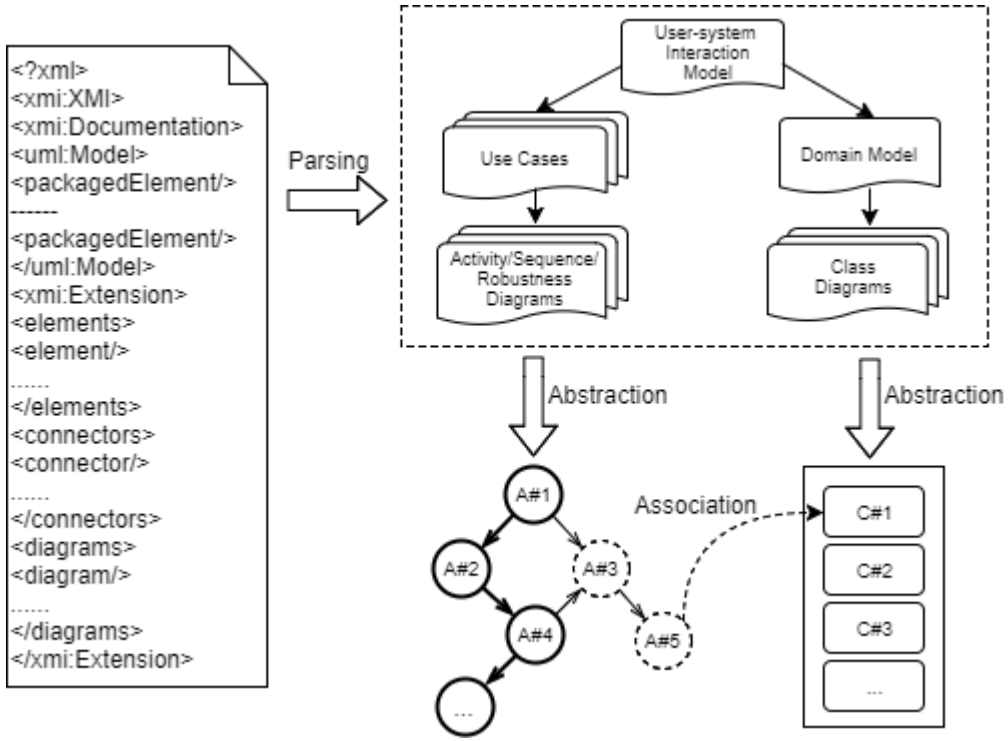
A set of notations are created to denote the key elements of USIM.



Automated Software Sizing Framework

The automated framework for software sizing.

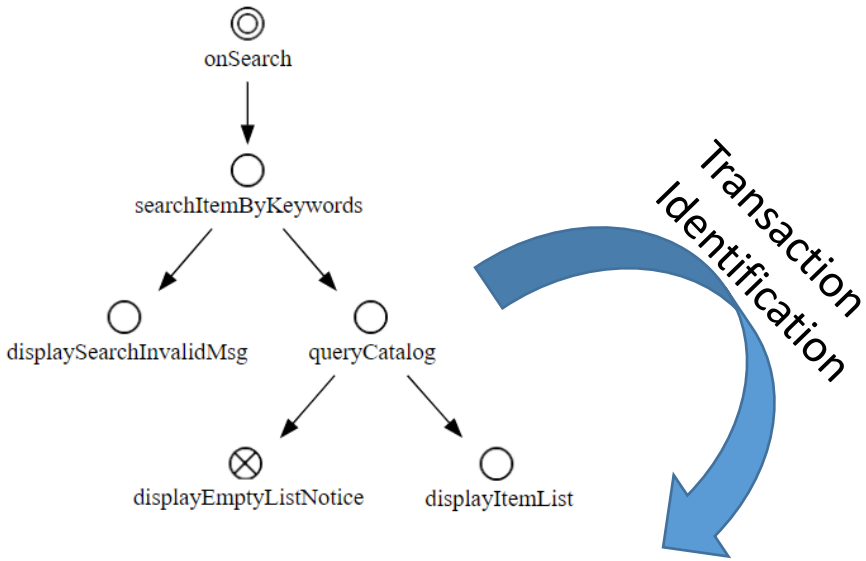
- XMI files exported from modeling tools are parsed to identify the UML elements. A hierarchy of UML elements is constructed based on the structure of XMI files.
- Model transformation algorithms are applied to construct the USIM.
- Automated transaction identification and classification procedures are applied to derive information for software size computation.



Automated Transaction Identification and Classification

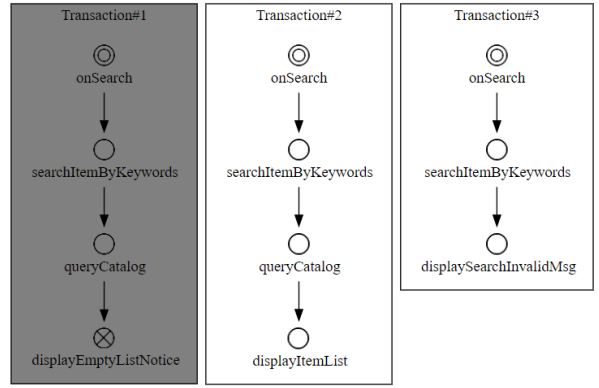
Transaction Identification

- Graph traversing algorithm (based on DFS) is applied to search for the paths from USIM. The paths are the transactions.
- Stimuli and system boundary define the entry and the exit conditions.
- System scope defines the effective transactions by excluding the transactions that are out of the scope.



Transaction classification

- Based on the properties of the identified transactions, for example, the number of activities (procedural complexity), the number of components (structural complexity), the number of associated data elements (data complexity)



Detailed Use Case Points (DUCPs)

DUCPs uses information derived from the automated transaction identification and classification processes, while it reuses the same approach to evaluating other project properties (UAW, EF, TCF) as the original Use Case Points (UCPs). The steps of counting DUCPs are as below:

1.Identify the transactions (T) from sequence diagrams for each use case of the system.
2.Count the transaction length (TL) and the data element types (DETs) for each identified transaction.
3.Determine the complexity level (CL) for each transaction by TABLE. 1 based on the TL and the DETs.
4.Determine the Unadjusted Detailed Transaction Weight (UDTW) for each transaction according to its CL and the weighting schema given in TABLE 2.
5.Calculate the Unadjusted Detailed Use Case Weight (UDUCW) by the equation below.

$$UDUCW = \sum_{c \in C} \sum_{t \in T_c} UDTW(t)$$

6.Evaluate the technical complexity factor (TCF) and the environmental factor (EF) based on the original UCPs.
7.Calculate the DUCPs by the equation below.

$$DUCP = (UDUCW + UAW) * TCF * EF$$

Table 1: Transactional Complexity Levels by DUCP

DET/TL	1-3	4-7	≥8
0-10	Very Low	Low	Medium
11-25	Low	Medium	High
≥26	Medium	High	Very High

Table 2: DUCP Weighting Schema

Complexity	Very Low	Low	Medium	High	Very High
UDTW	1	2	5	13	21

Model Calibration

Data Collection

- 22 projects from 577 and 590 (2014-2017).
- Sequence and class diagrams were collected.
- Effort data were collected by weekly reports and Jira.

Model Calibration

- Exploratory data analysis was applied to determine the linear relationship and its significance. The calibrated model is as follows:

Table 3: SE, T-values, P-values for the Parameters

Parameter	est.	std. error	t-value	p-value
β_0	11.57	147.37	0.08	0.94
β_1	3.14	0.50	6.33	0.36e-05

Model Evaluation

- Evaluated out-of-sample estimation accuracy by 5-fold cross validation in terms of MMRE, PRED(.15), PRED(.25), PRED(.50).
- In the comparison with the original UCPs, DUCPs is about 21% better for MMRE, 14% better for PRED(.15), 14% better for PRED(.25), and 15% better for PRED(.50).

Table 4: MMRE, PRED(.15), PRED(.25), PRED(.50) for 5-fold Cross Validation

	DUCPs				UCPs			
Run	M.	P.(.15)	P.(.25)	P.(.50)	M.	P.(.15)	P.(.25)	P.(.50)
1	0.45	0.00	0.20	0.60	0.61	0.00	0.20	0.60
2	0.19	0.25	0.75	1.00	0.56	0.00	0.50	1.50
3	0.30	0.25	0.25	1.00	0.57	0.00	0.00	0.75
4	0.37	0.25	0.50	0.50	0.59	0.25	0.50	0.50
5	0.21	0.20	0.60	1.00	0.28	0.00	0.40	1.00
Avg.	0.31	0.19	0.46	0.82	0.52	0.05	0.32	0.67

Conclusions

- Proposed the framework to automate software sizing based on class and sequence diagrams.
- Formally defined the user-system interaction model (USIM) that bridges the gap between software sizing model and UML metamodels.
- Proposed a software size metric (DUCPs) that can be automated.
- Validated the effectiveness of DUCPs in effort estimation by an empirical study of 22 projects and the comparison with original UCPs.

Future Directions

- Collect more data points from different software engineering settings to further prove the performance of DUCPs and its significance.
- Integrate other types of UML diagrams into the proposed evaluation paradigm.

Thank you!

If you would like to contribute any project data (use case driven projects, UML-based projects, or others) to the research, please contact me at : kqi@usc.edu.

Q&A

References

[1] Thomas Fetcke, Alain Abran, and Tho-Hau Nguyen. Mapping the oo-Jacobson approach to function point analysis. In Software Metrics, pages 59–73. Springer, 1997.

[2] Takuya Uemura, Shinji Kusumoto, and Katsuro Inoue. Function point measurement tool for uml design specification. In Software Metrics Symposium, 1999. Proceedings. Sixth International, pages 62–69. IEEE, 1999.

[3] SangEun Kim, William M Lively, and Dick B Simmons. An effort estimationby uml points in early stage of software development. InSoftware EngineeringResearch and Practice, pages 415–421, 2006.

[4] Arlene Minkiewicz. Measuring object oriented software with predictive objectpoints.PRICE Systems, LLC, 1997.

[5] Gennaro Costagliola, Filomena Ferrucci, Genoveffa Tortora, and Giuliana Vitiello.Class point: an approach for the size estimation of object-oriented systems.IEEETransactions on Software Engineering, 31(1):52–74, 2005.

[6] Massimo Carbone and Giuseppe Santucci. Fast&&serious: a uml based metricfor effort estimation. InProceedings of the 6th ECOOP workshop on quantitativeapproaches in object-oriented software engineering (QAOOSE'02), pages 313–322,2002.

[7] Yue Chen, Barry W Boehm, Ray Madachy, and Ricardo Valerdi. An empiricalstudy of eservices product uml sizing metrics. InEmpirical Software Engineering,2004. ISESE'04. Proceedings. 2004 International Symposium on, pages 199–206. IEEE,2004.