Appendix -1
Architectures for Software Systems
NBU Master's program

Courses Objectives:
Architectures for Software Systems aims to teach you how to design, understand, and evaluate systems at an architectural level of abstraction. The course is developed by Carnegie Mellon University, Institute for Software Research and is delivered under the supervision of European Software Institute (ESI).

By the end of the course you will be able to:

- Understand the influence of architectural drivers on software structures.
- Understand the technical, organizational, and business role of software architecture.
- Identify key architectural structures (styles, patterns, tactics, etc.).
- Understand the principles of good architectural documentation and presentation.
- Understand the impact that COTS has on architectural designs.
- Generate architectural alternatives in a given context and choose among them.
- Understand how formal notations can be used to specify architectures.
- Evaluate the fitness of an architectural design in meeting a set of system requirements and balancing quality tradeoffs.
- Be aware of the future trends in software architecture.

Course Organization:
Lectures: The original CMU course has been split to **Architecture for Software Systems 1 - ESMM290 (fall)** and **Architecture for Software Systems 2 ESMM390 (spring)** courses.
Each course has 15 lectures and takes 30 astronomic hours. There will be **one lecture** weekly. Note that you have to subscribe to both courses and both recitations. You can drop after the first semester but you are not allowed to subscribe just to the second course. Attendance is mandatory.

Recitations: To help you master key techniques and tools, provide guidance for assignments, and review the results of assignments, there will be **one recitation** weekly. During the fall you have to attend to **ESMM292** and during the spring to **ESMM392.** For some recitations, the class will be divided into smaller groups. Attendance is mandatory.

Guest Lectures:

On each of the courses during the recitations there will be 2 guest lectures from Leading Bulgarian Companies that will present to you the role of the Architecture in their organizations and products.

Bootcamp: During the second semester there will be a bootcamp event that will give you a job market orientation and will train you how to sell yourself better on the Bulgarian job market.

Computing: A personal computer or laptop is required for both courses and recitations. For the course assignments we will be using the Java programming language. You will need to download and install the Java Software Development Kit (the latest version of J2SE). You may use any editor or programming environment that you like.

Communication: Course materials are available through NBU Moodle System: http://www.e-edu.nbu.bg

Textbooks:

- *Software Architecture: Perspectives on an Emerging Discipline*, by Mary Shaw and David Garlan, Prentice Hall 1996 [SG96]
- *Software Architecture in Practice, Second Edition*, by Len Bass, Paul Clements, and Rick Kazman, Addison-Wesley 2003 [BCK03].
- *Documenting Software Architectures: Views and Beyond, Second Edition,*by Clements, et al. Addison-Wesley 2011 [C+11].
- *Architecting Software Intensive Systems: A Practitioner's Guide,*by Anthony J. Lattanze, Taylor and Francis/Auerbach 2008 [ASIS08].

We will also use a collection of supplementary readings, which will be available through the course on the e-edu.nbu.bg web site.

Grading:

Your SA1 and SA2 course grades will be determined as a combination of four factors:

- Questions on Readings: (30%) Each week's lectures will be accompanied by one or more readings, which we expect you to read before you come to the first class of the week. To help you focus your thoughts on the main points of the readings, we will assign one to five questions (based on the readings) to be answered each week. Each question should be addressed in less than a page (often a single paragraph will do), and is due at the first lecture in the week for which it is assigned. All solutions must be submitted through e-edu.nbu.bg. Late submissions will not be accepted unless you make prior arrangements with the instructor – each request will be handled on a case-by-case basis.
- Assignments: (40%) There will be six assignments (3 in **Recitations 1** and 3 in **Recitations 2**). Some of the recitations will be a small system architecting exercises some will be recovering and analysis of an existing system or standard. Each assignment will have a group and an individual section. The purpose of these assignments is to give you some experience analyzing architectures and the systemic properties they possess. To help clarify your designs we will hold a brief, ungraded design review for most assignments during class a week before it is due. Groups will take turns

presenting their preliminary designs and getting feedback from the class and instructor. Late assignments may be accepted, but will be steeply penalized.

- Final Project: (30%) The course project is designed to give you experience with the architecture of a larger software system. You will design and analyze the architecture of a system, document your work, and present the results to the rest of the class. You will be evaluated on both the quality of your design, analysis and documentation, as well as on your final presentation. *Note that final project presentations will take place during the first week of exams at the normal recitations lecture time. Students are required to be present for ALL presentations.*
- Instructors' judgment: The instructor reserve the right to raise or lower your quantitatively-determined grade based on their judgment of your mastery of course material; this judgment will be based in part on your ability to participate constructively in class discussions. Participation in labs and recitations may also be taken into consideration.

Late Assignment Policy: In general, late answers to the weekly reading questions will not be accepted without prior approval. Late team-based assignments will be penalized. If a personal emergency should arise that affects your ability to turn in an assignment in a timely fashion, you must contact the course instructor as soon as possible.

Cooperation Policy and Quotations: We encourage vigorous discussion and cooperation in this class. You should feel free to discuss any aspects of the class with any classmates. However, we insist that any written material that is not specifically designated as a Team Deliverable be done be you alone. This includes answers to reading questions, individual reports associated with assignments, and labs. We also insist that if you include verbatim text from any source, you clearly indicate it using standard conventions of quotation or indentation and a note to indicate the source.

Syllabus

Architecture for Software Systems 1 (ESMM290)

Lecture	Date	Major Topic	Subtopic	Readings
Software Architecture 1				
1		Introduction	Course Overview	
2			Software Architecture Defined	SG96 Ch 1 (skim); BCK03 Ch 2 (skim); ASIS08 Ch 2 (skim)
3			Enterprise, System and Software architecture	ASIS08
4			The Business of Architecture	ASIS08
5			Architectural Drivers	BCK03 Ch 4; ASIS08 Ch 3

6		Structures	Structures and Views	SG96 Ch 2; ASIS08 Ch 4
7			Documentation (Part 1)	C+11 Prologue, Ch 1, 3
8		Dataflow Systems	Dataflow Styles	SG96 3.1, 3.2, 4.2.1, and 4.3.1
9			Case Study	Lat99
10		Event-Based Systems	Event Styles	SG96 Ch 2.4, 2.6, 2.7, 7.3
11		Call-Return Systems	Call-Return Styles	Par72; C+11 2.4; Dvo02
12			Client-Server and Tiered Architectures	STR00
13			Middleware	Vin02
14		Shared Info Systems	Shared Information Styles	Nii86; Ext09; Sum04
15		Techniques and Methods	Guidance for the Architect	ASIS Ch 5

Architecture for Software Systems 2 (ESMM390)

1			Design by Selection	The Edge, Vol 2, No 1, Mitre Corp, March 2001
2			Architecture Evaluation	BCK03 Ch 11; JL01 Entire Report
3			Strategic Reuse	BCK03 Ch 14, 15
4			Strategic Reuse Case Studies	SG96 3.2; Gar+05
5			Platform Case Studies	MF93; BW08, S+11
6			Architectures, Lifecycles, and Process – part 1	Nor98; PLK07
7			Architectures, Lifecycles, and Process – part 2	Blu03; Lat12
8			Documentation 2: Using UML for Design Representation	RJB04; I+04; UML Tool
9			Architecture Conformance	S+06; BCK03 Ch 10
10		Architecting	SOA and Web Services	Erd09; Foo99;

		for X		Obrien05
11			Cloud Computing Case Studies	Liu12, Risten09
12			Formal Specification and Analysis	SG96 3.2, 6.1-6.3,8.3, GS06
13			Security & Performance	BIT+10, SG90; Mul10
14			Availability	SG96 6.1-6.3, 8.3; GS06
15			Research Directions	G+02; GS07; GSC09

Readings

AK01, Anita King, Commercial Off-the-Shelf Software - Benefits and Burdens, The Edge, Volume 2, Number 1, Mitre Corporation, March 2001.

ASIS08, Architecting Software Intensive Systems: A Practitioner's Guide, by Anthony J. Lattanze, Taylor and Francis/Auerbach 2008 [Lat08].

BCK03, Software Architecture in Practice, Second Edition, by Len Bass, Paul Clements, and Rick Kazman, Addison-Wesley 2003 [BCK03].

BIT+10, Terry V. Benzel, Cynthia E. Irvine, Timothy E. Levin, Ganesha Bhaskara,Thuy D. Nguyen, and Paul C. Clark, Design Principles for Security, Naval Post Graduate School Report (NPS-CS-05-010), USC Information Sciences Institute Report (ISI-TR-605)

Blu03, J. Blunt, A Framework Does Not an Architecture Make, InfoEd Experience White Paper, Info Ed TiAC White Paper Repository, 2003

BW08, Carliss Y. Baldwin and C. Jason Woodard, The Architecture of Platforms: A Unified View, 2008.

C+11, Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford. Documenting Software Architectures: Views and Beyond, Second Edition. Addison-Wesley 2011. Prologue, Chapters 1, 3.

Dvo02, Challenging Encapsulation in the Design of High-Risk Control Systems by Daniel Dvorak. 17th Annual ACM Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA 2002), November 2002.

Erd09, Hakan Erdogmus, Cloud Computing: Does Nirvana Hide behind the Nebula? IEEE Software, March/April 2009

Ext09, Extropia: Introduction to Databases for Web Developers (Part 1): http://www.extropia.com/tutorials/sql/toc.html

Foo99, Brian Foote and Joseph Yoder, Big Ball of Mud, Fourth Conference on Patterns Languages of Programs, 1999.

G+02, David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste. Project Aura: Towards Distraction-Free Pervasive Computing. IEEE Pervasive Computing, special issue on "Integrated Pervasive Computing Environments", Volume 21, Number 2, April-June, 2002. pp. 22-31.

Gar+05, Bridging the Gap between Systems Design and Space Systems Software David Garlan, William K. Reinholtz, Bradley Schmerl, Nicholas Sherman and Tony Tseng. In Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop (SEW-29), Greenbelt, MD, 6-7 April 2005.

GCS03, David Garlan, Shang-Wen Cheng, and Bradley Schmerl, "Increasing System Dependability through Architecture-based Self-repair". In Architecting Dependable Systems, Springer-Verlag, 2003.

GS06, Architecture-driven Modelling and Analysis, David Garlan and Bradley Schmerl. In Tony Cant editor, Proceedings of the 11th Australian Workshop on Safety Related Programmable Systems (SCS'06), Vol. 69 of Conferences in Research and Practice in Information Technology, Melbourne, Australia, 2006.

GS07, David Garlan and Bradley Schmerl, "The RADAR Architecture for Personal Cognitive Assistance", International Journal of Software Engineering and Knowledge Engineering, Vol. 17(2), April 2007.

GSC09, David Garlan, Bradley Schmerl, and Shang-Wen Cheng, “Software Architecture-Based Self-Adaptation”. In Autonomic Computing and Networking, Springer Verlag, 2009. (To appear)

I+04, Sections 3-5 of "Documenting Component and Connector Views with UML 2.0", CMU/SEI-2004-TR-008.

IBM05, Capturing Architectural Requirements, Peter Eles, Senior IT Architect IBM
http://www.ibm.com/developerworks/rational/library/4706.html

JC01, Judy Clapp, Understanding Risks Alleviates COTS-based Systems Woes, The Edge, Volume 2, Number 1, Mitre Corporation, March 2001.

JL01, Lawrence G. Jones and Anthony J. Lattanze, "Using the Architecture Tradeoff Analysis Method to Evaluate a Wargame Simulation System: A Case Study", December 2001, CMU/SEI-2001-TN-022.

KB+11, Karyn Benson, Rafael Dowsley, Hovav Shacham, Do You Know Where Your Cloud Files Are? CCSW 11, October 2011, ACM 978-1-4503-1004-8/11/10

Lat12,A. Lattanze, Infusing Architectural Thinking into Organizations, IEEE Software Jan/Feb, 2012

Liu12, CH_16a_Cloud_Deployment_AnnaLiu.pdf (172.63 KB)

Lat99, Case Study: An Early Exploitation of Product Lines by Anthony J. Lattanze. Proceedings of the International Society for Computers and Their Applications (ISCA) 1st International Conference on Information Reuse and Integration (IRI-99), Atlanta, Georgia, USA, November 4-6, 1999.

MF93, Charles R. Morris and Charles H. Ferguson, How Architecture Wins Technology Wars, Harvard Business Review Reprint 93203.

MS+06, Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, Peter Sommerlad, "Security Patterns: Integrating Security and Systems Engineering", Sections 6.4-6.5, 2006

Mul10, Introduction to System Performance Design, Gerrit Muller, January 2010, draft.

Nii86, H. Penny Nii. Blackboard Systems. AI Magazine 7(3):38-53 and 7(4):82-107.

Nor98, Donald A. Norman, The Invisible Computer: Why Good Products Can Fail, the Personal Computer is So Complex, and Information Appliances Are the Solution, Chapter 2, 1998.

OBrien05, Liam O'Brien, Len Bass, Paulo Merson, Quality Attributes and Service-Oriented Architectures, CMU/SEI-2005-TN-014, September 2005

Par72, On the Criteria To Be Used in Decomposing Systems Into Modules by David L. Parnas. Communications of the ACM, 15(12):1053-1058, December 1972.

PLK07, Jinalben Patel, Roger Lee, and Haeng-Kon Kim, Architectural View in Software Development Life-Cycle Practices, 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), IEEE Computer Science, 0-7695-2841-4/07, 2007.

RJB04, J. Rumbaugh, I. Jacobson, G. Booch, /The Unified Modeling Language Reference Manual/, /Second Edition/, Addison Wesley, 2004. Chapter 3.

S+06, Bradley Schmerl, Jonathan Aldrich, David Garlan, Rick Kamzman, and Hong Yan, "Discovering Architectures from Running Systems, IEEE Transactions on Software Engineering, VOl. 32, No. 7, July 2006.

S+11, Bradley Schmerl, David Garlan, Vishal Dwivedi, Michael W. Bigrigg, and Kathleen M. Carley, SORASCS: A Case Study in SOA-based Platform Design for Socio-Cultural Analysis, 2011.

SG90, L. Sha, J. Goodenough, "Real-Time Scheduling Theory and Ada", IEEE Computer, April 1990, 53-62.

SG96, Software Architecture: Perspectives on an Emerging Discipline, by Mary Shaw and David Garlan, Prentice Hall 1996 [SG96]

STR00, Three Tier Software Architectures, Software Technology Review, CMU SEI 16-Feb-2000.

Sum04, William Sumner, An Introduction to Shared Memory, Hewlett Packard

Risten09, Thomas Ristenpart, Eran Tromer, Hovav Shacham, Stefan Savage, Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds, CCS 09, November 2009, ACM 978-1-60558-352-5/09/11

UML Tool, http://en.wikipedia.org/wiki/UML_Tool

Vin02, Steve Vinoski, "Where is Middleware?", IEEE Internet Computing, March/April 2002, Vol 2, Issue 2, pp. 83-85.

Yod97, Joseph Yoder, Jeffery Barcalow, Architectural Patterns for Enabling Application Security, PLoP 1997