УДК 502:004.45 (075.8) https://doi.org/10.15407/pp2018.04.030

Н.А. Сидоров

50 ЛЕТ ИНЖЕНЕРИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Статья посвящается 50-ой годовщине образования, ключевой на сегодняшний день области информатики – инженерии программного обеспечения. В октябре 2018 года исполнилось 50 лет со дня проведения конференции, на которой профессиональное сообщество программистов и ученых ввело в обращение и обосновало термин software engineering. Статья основана на сорока пятилетнем опыте автора в инженерии программного обеспечения и цель статьи двоякая. С одной стороны, отметить важную дату для главной отрасли страны – индустрии программного обеспечения, а с другой – сделать обзор состояния дел в инженерии программного обеспечения, отметив тех, кто внес системообразующий научный вклад в развитие отрасли. Статья состоит из трех разделов. В первом, излагаются истоки и условия, которые привели к появлению инженерии программного обеспечения. Во втором, приводятся системообразующие результаты и указываются авторы этих результатов. При этом, рассматриваются такие разделы инженерии программного обеспечения как программирование, в аспекте структурного операторного базиса современных языков программирования; модуляризация как основа для повторного и многократного использования программного обеспечения; моделирование жизненного цикла, важность которого состоит в том, что оно не только привело к управлению жизненным циклом, но открывая новые процессы, определило новые продукты и ресурсы необходимые для реализации этих процессов и позволило перейти к «программированию в большом», что потребовало создания новых методов, инструментов и профессий; эмпирическая инженерия программного обеспечения, на сегодня, это раздел инженерии программного обеспечения, представленный большим количеством и разнообразием метрик, инструментами и методами проведения измерений и анализа результатов; культура программного обеспечения, которая утверждает, что создавать качественное и надежное программное обеспечение могут только коллективы, обладающие определенной культурой и зрелостью; экономика программного обеспечения и модели для оценки стоимости программного обеспечения; зеленые информационные технологии и программное обеспечение, экосистемы. В третьем разделе, рассматривается постановка образования в инженерии программного обеспечения.

Ключевые слова: программирование, программное обеспечение, инженерия программного обеспечения, обучение.

Введение

Статья посвящается 50-ой годовщине образования, ключевой на сегодняшний день области информатики – инженерии программного обеспечения. В октябре 2018 года исполнилось 50 лет со дня проведения конференции, на которой профессиональное сообщество ученых ввело в обращение и обосновало термин software engineering [1], названный А.П. Ершовым в русском варианте – технология программирования [2].

Значительно позднее этот термин стали использовать как программная инженерия, в Украине – с 2006 и до 2016 года, а в России используется до настоящего времени. В 2016 году в Украине, учитывая подготовку новых стандартов обучения, удалось привести термин в соответствие с английским. Теперь, это инженерия программного обеспечения. Процесс оказался длительным и болезненным, по двум причинам. Во-первых, из-за упрощенного взгляда на создание и сопровождение программ как программирование, которое и сегодня существует в промышленности, вследствие того, что распространено оффшорное производство, содержание которого сводится к кодированию. Во-вторых, из-за отсутствия значительной массы специалистов как в промышленности, так и в высшей школе, которые бы хорошо понимали предмет. Например, в 2006 году когда решался вопрос о содержании учебного плана, создаваемого бакалаврата Программная инженерия, то пришлось преодолевать значительное сопротивление сторонников компьютерных наук. Однако, несмотря на то, что правильный план был принят [3], в большинстве университетов до сегодняшнего дня читается то, что

«может читаться», а не то, что нужно читать по плану. Главная причина этого – катастрофическая нехватка специалистов по дисциплинам инженерии программного обеспечения, которые так и остаются новыми для большинства преподавателей.

Поэтому у статьи две цели. Одна, отметить важную дату для главной отрасли страны – индустрии программного обеспечения. С другой – дать полную картину состояния дел инженерии программного обеспечения, помянув выдающихся личностей, кто внес системообразующий вклад в развитие отрасли. Учитывая данное нужно отметить, что автор, во-первых, излагает собственный, хотя, по возможности и аргументированный взгляд на роль тех или иных личностей в истории инженерии программного обеспечения. Во-вторых, руководствовался только значением результатов этих личностей для развития отрасли, принципиальным влиянием результатов на ход развития инженерии программного обеспечения.

Материал изложен в такой последовательности. Вначале рассматривается прошлое отрасли, затем называются выдающиеся личности и их результаты, и, наконец автор рассматривает проблемы образования и в Украине в инженерии программного обеспечения.

Материал статьи докладывался на конференции УкрПрог2018, но не был опубликован.

Начало инженерии программного обеспечения

Вследствие основания Ч. Бэббиджем, функционирования своей машины на принципе программного управления появилась необходимость писать программы, появился процесс – программирование и первые программисты (*А. Лавлейс*). Впоследствии, с появлением электронных вычислительных машин (1947, Великобритания – М. Уилкс; 1950, СССР – С. Лебедев) и их широким распространением, этот процесс стал массовым, а, принимая во внимание его сложность и стоимость писателей и исполнителей программ еще и дорогим. Б. Боэм вспоминает [4], что, когда он в 1955 году пришел писать программы в General Dynamics, преобладало мнение, что инженер программист подобен инженеру по «железу». Все, кто работал в General Dynamics был либо инженер по «железу», либо математик. Поэтому, прежде чем выполнять свой код на компьютере все пользовались «железным» принципом – много раз отмерь, один раз отрежь. Кроме этого, Б. Боэм вспоминает, что в первый день начальник, показывая ему компьютер, который занимал огромную комнату, напоминал, что за этот компьютер General Dynamics платит 600 долларов в час и еще Боэму 2 доллара, и что он хочет, чтобы Б. Боэм действовал адекватно, чаще практикуя отладку за столом, парную отладку и ручное исполнение программы. Такой взгляд, как оказалось позже, не был плодотворным, однако благодаря этому взгляду возникли такие организации как *Association on Computing Machinery и IEEE Computer Society*. Надо отметить, что такая же ситуация складывалась и в Советском союзе после ввода в эксплуатацию МЭСМ и БЭСМ [5].

К 1960 году пришло понимание, что программное обеспечение принципиально отличается от «железа». Во-первых, программное обеспечение легко модифицировалось и копировалось. Во-вторых, программное обеспечение не изнашивалось, а его сопровождение отличалось от сопровождения «железа». Программное обеспечение было невидимо, не имело веса, но при этом было очень дорогим. Разработка его была практически неуправляемой и требовала огромного количества спецификаций (в 50 раз больше чем для «железа» [6]). В-третьих, с распространением машин в программировании появилась кадровая проблема – нехватка программистов, которую решали путем тренинга гуманитариев, филологов, социологов и специалистов из подобных отраслей. Этим людям было комфортно в «code-and-fix» модели жизненного цикла, что вело к производству «спагетти» кода и развитию «hacker» культуры, воспитывающей программистов «ковбоев». Однако, не все в 60-х годах поддавались влиянию «code-and-fix» модели. Отличной от этой и характерной, для

того времени, с точки зрения инженерии программного обеспечения была позиция менеджмента производства операционной системы *OS-360* (*F.P. Brooks* [7]). Но, в целом, ситуация характеризовалась следующим образом:

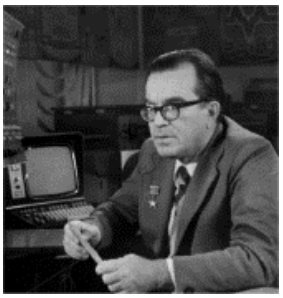
- в основном ручная работа (software crafting), поэтому дорого и ненадежно;
- преобладание модели «code-and-fix», поэтому запутанный код;
- “heroic debugging”, поэтому много дефектов;
- большие проекты, но «слабые» планирование и управление, поэтому срывы поставок;
- милитаризация, требование роста, разнообразия и сопровождения;
- нехватка умений и кадров.

Это был кризис. Для выхода из кризиса стали искать пути – «серебряную пулю» [8]. Эти поиски привели к необходимости обсуждения накопившихся проблем и организации конференции. Председателем конференции был выбран профессор *F.L. Bauer.* Подготовка к конференции была начата весной 1968 года, а в октябре конференция состоялась [1]. Был введен термин *Software engineering* – приложение систематического, дисциплинированного, измеримого подхода к разработке, функционированию и сопровождению программного обеспечения, а также исследованию этого подхода; приложение дисциплины инженерии к программному обеспечению (ISO/IEC/IEEE 24765-2010). Термин был выбран специально «провокационный». Дискуссии были посвящены всем аспектам программного обеспечения, включая связь программного и аппаратного обеспечения компьютера, проектирование и производство программного обеспечения, сопровождение и обслуживание программного обеспечения, специфицирование и управление большими проектами, образование и подготовка инженеров по программному обеспечению, экономика [1].

По результатам конференции были изданы известные материалы, подготовленные *B. Randell* и *P. Naur* [1]. В материалах рассматривалось содержание термина инженерия программного обеспечения (software engineering). Конференция обеспечила понимание следующего:

- состояние теории и практики индустрии программного обеспечения как основы для определения и разработки улучшений;
- необходимость в новых методах и практиках для создания и сопровождения программных продуктов;
- необходимость в специальной подготовке кадров.

Таким образом были начаты работы в новой области науки и техники – инженерии программного обеспечения. На сегодня, это большая, эффективная и результативная часть информатики. Например, только в Украине работает около 200 компаний с численностью сотрудников от 80 до 800+, Software outsourcing export составлял $ 2.5 млрд. (2015), а в 2017 году отрасль выросла на 27 %, Freelancers зарабатывали $ 60 млн. (2014); сейчас работает 126990 специалистов, а к 2020 году будут требоваться 200 000 [9 – 11].

А.П. Ершов, являясь членом ТК 2 IFIP, ввел в Советском союзе термин технология программирования. В Украине основные работы по программному обеспечению были сосредоточены в Институте кибернетики АН УССР, а руководил ими В.М. Глушков. Кроме В.М. Глушкова ключевыми учеными в технологии программирования были А.П. Ершов, который ввел три инженерных подхода к созданию программного обеспечения – конкретизирующее, синтезирующее и сборочное про-


граммирование [12]; И.В. Вельбицкий создавал теоретические и практические основы для индустриального производства программного обеспечения (Стандартизованные Элементы Языка, Р – технология программирования [13–15]); В.В. Липаев исследовал надежность и качество программного обеспечения, внедрял стандартизацию [16, 17]. В 1979 году, в Киеве состоялась I Всесоюзная конференция «Технология программирования», которую готовил И.В. Вельбицкий, а в 1985 году уже в Калинине (Тверь) состоялась Всесоюзная научно-техническая конференция «Программные средства как продукция производственно-технического назначения».

Выдающиеся личности и результаты

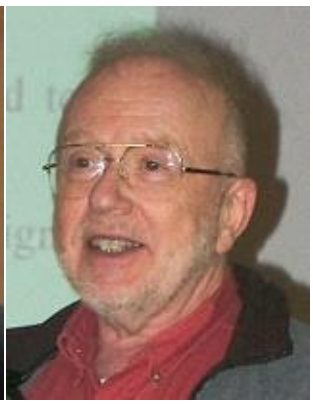
Программирование. Поскольку основной процесс в инженерии программного обеспечения – это программирование, то первые результаты были получены в данной области. Конечно, это структурное программирование [18, 19], разработка которого *E.W. Dijkstra* на основе известной


структурной теоремы [20] *C. Bohm* и *G. Jacopini*, позволило аргументировано отказаться от использования оператора *go to* и обеспечило реальный путь к созданию понятных программ.

Модуляризация. В первом опыте появилось понимание того, что программы, также как изделия других отраслей, должны строиться из готовых блоков. Поэтому следующие результаты были получены в области модуляризации программ, то есть представления программ из частей, модулей. Первым получил результаты в этой области *M.V. Wilkes* для первой электронной вычислительной машины, работающей на принципах *J. von Neumann*, которую в 1947 году он построил, была написана операционная система, обеспечивающая обработку подпрограмм [21]. Понятие подпро-


граммы, хотя и использовалось только для уменьшения рутинной работы в процессе программирования, стало первым средством модульного представления программ.

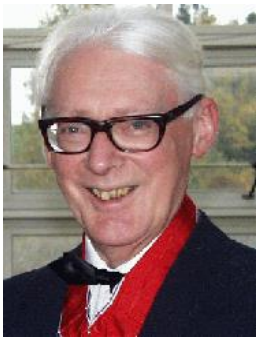
Следующим шагом в этом направлении было введение *K. Samelson,* в 1959 году для языка *Algol* понятия блока [22]. Блок, кроме указания границ модуля обеспечил регулирование области действия и времени существования объектов (переменных), описанных в нем. Позднее блок и подпрограмма составили основу блок-ориентированных языков. Однако оставался открытым вопрос, относящийся к определению границ и размеров модуля. Для ответа на этот вопрос *L.L. Constantine*

в 1968 году ввел понятия связывания (*cohesion)* частей, составляющих модуль и соединения (*coupling)* для указания соединения между модулями [23]. В 1972 году *D.L. Parnas* ввел конкретные критерии модуляризации и предложил устройство модуля на основе понятия сокрытия информации [24]. Модуль должен был состоять из двух частей. Одна часть – definition, должна содержать описание ресурсов, которые предоставляет модуль, а вторая часть – implementation, содержит реализацию этих ресурсов. Таким образом, *D.L. Parnas* реализовал сокрытие информации, обеспечив закрытость реализации

модуля и его независимость. Это открывало возможность реализации компонентов многократного использования. В 1980 году *N. Wirth* реализовал язык программирования *Modula*, а позже *Modula-2*, в которых использовалось понятие модуля, пред-

ложенное *D.L. Parnas* [24]. В 1984 году по проекту *J. Ichbiah* был создан язык программирования *Ada*, в котором такое же понятие реализовано в форме пакета [25]. В 1967 году *O.-J. Dahl* и *K. Nygaard* использовали блок и сокрытие информации при разработке языка *Simula 67*, в котором были заложены основы объектно-ориентированных языков [26]. Эти основы




получили развитие благодаря работам *C.A.R. Hoare* 1969 года по концепциям наследования, позднего связывания и работам *N. Wirth no* ссылкам [26, 27]. Завершены эти работы в области объектно-ориентированных языков были в 1970 году в «чистом» объектно-ориентированном

языке *SmallTalk A. Kay* [28]. Таким образом, в основном были завершены работы в области модуляризации, как для композиционных, так и классификационных языков и создана основа для повторного и многократного использования программ-

ного обеспечения. В настоящее время эти работы развиваются в направлении исследования и создания программного обеспечения как системы систем (systems of systems), используя связь системного анализа и инженерии программного обеспечения, и развивая системную инженерию программного обеспечения (system software engineering) [29].

Жизненный цикл. Системообразующие для инженерии программного обеспечения результаты были получены в управлении процессами создания и сопровождения программного обеспечения – это результаты по моделированию жизненного цикла. Важность их состоит в том, что они не только привели к управлению жизненным циклом, но открывая новые процессы, определяли новые продукты и ресурсы необходимые для реализации этих процессов. Это привело к уходу от «программирования в малом» и переходу к «программированию в большом», что потребовало создания новых методов, инструментов и профессий. Поэтому жизненный цикл играет в инженерии программного обеспечения системообразующую роль. Фундаментальные результаты были получены *W.W. Royce*, который в 1970 году предло-

жил, а в 1978 году уточнил каскадную модель жизненного цикла [30]. Таким образом, один процесс в модели *«Code-end-fix»* представлялся шестью процессами в модели *W.W. Royce «Waterfall»*. И, хотя, как оказалось позднее, каскадная модель не имела практического значения, она, также как в свое время язык программирования *Algol 58*, стала фундаментальной основой для всех моделей жизненного цикла, которые появились позже нее. Другой моделью, которая уточняла каскадную модель в практическом аспекте – спиральная, была

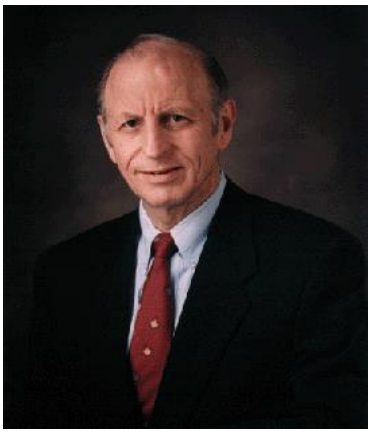
предложена в 1988 году *B.W. Boehm* [31]. Сейчас известно более тридцати моделей жизненного цикла, которые так или иначе основываются на этих двух моделях [32].

Важную роль в контексте жизненного цикла сыграли работы *M. Lehman*, который указал на три типа программ: S- (specification), P- (problem) и E- (environment). Он показал, что сопровождение программного обеспечения следует рассматривать как эволюцию, ввел математическую модель и законы эволюции [33]. Таким образом, открыв взгляд на сопровождение программного обеспечения как на разработку и создал основу для повторного использования программного обеспечения (software reusing). В 1980 году *J.M. Neighbors* защитил диссертацию по повторному использованию, а в 1984 году под его редакцией вышел тематический номер журнала IEEE Transactions on Software engineering, который был посвящен повторному использованию программного обеспечения [34, 35]. *B.W. Boehm* в статье посвященной путям повышения эффективности разработки программного обеспечения, назвал повторное использование наиболее перспективным направлением [36]. В этой связи, получили развитие три связанные области, образующие в целом утилизацию программного обеспечения, это повторное использование (reuse), восстановление (recovery, reability) и переработка (rework) наследуемого (legacy software) программного обеспечения [37]. Первые результаты

в этом направлении получили *T. Biggerstaff* и *R. Prietto-Diaz*. Работы по утилизации программного обеспечения требовали не только анализа наследуемого программного обеспечения в контексте процессов его разработки, но более широкого анализа предметной области и построения нового программного обеспечения на основе наследуемого. Это привело к появлению реверсивной инженерии программного обеспечения (reverse engineering), доменного анализа (domain analysis) и реинженерии (reengineering) [38–42]. Таким образом, в контексте инженерии программного обеспечения, кроме прямой инженерии (forward engineering), направленной на создание программных продуктов начали исследоваться и применяться еще две инженерии – обратная (backward engineering) или реверсивная инженерия, направленная на восстановление информации о наследуемом программном обеспечении, и реинженерия, направленная на переработку наследуемого программного обеспечения [43].

Эмпирическая инженерия программного обеспечения. Очевидно, так как создание и сопровождение программного обеспечения – это инженерная деятельность, то она не могла быть эффективной без количественной оценки свойств и характеристик составляющих жизненного цикла – процессов, продуктов и ресурсов. В основе этой оценки в инженерных отраслях лежит измерение. Поэтому, указанные инженерии в 2005 году благодаря работам *B.W. Boehm* и *V.R. Basili* были дополнены еще одной – эмпирической инженерией программного обеспечения (software empirical engineering) [44]. Сегодня, это большой раздел инженерии программного обеспечения, представленный значительным количеством и разнообразием метрик, инструментами и методами проведения измерений и анализа результатов [45, 46].

Культура программного обеспечения. Как рано выяснилось – создание и сопровождение программного обеспечения, это коллективная деятельность, а сейчас еще и глобально распределенная. Поэтому, создавать качественное и надежное программное обеспечение могли только коллективы, при этом, обладающие определенной культурой и зрелостью. В 2001 году *L.L. Constantine* ввел понятие культуры программного обеспечения (software culture), рассматривая командную разработку и парадигмы культуры [47]. В 1989 году *W.S. Humphrey* разработал модель для оценки зрелости процессов разработки программного обеспечения *Capability Maturity Model (CMM*), которая позволила оценивать в том числе и культуру программного обеспечения [48, 49]. В 2005 году в *Software engineering institute* была разработана интегрированная версия моделей *CMM – Capability Maturity Model Integration (CMMI*) [49]. В 1999 году *ACM* и *IEEE Computer Society* был разработан кодекс этики и профессиональной практики инженера по программному обеспече-



нию (*Software Engineering Code of Ethics and Professional Practice)* [51, 52].

Экономика программного обеспечения. Проблемы экономики программного обеспечения ставились уже на первой конференции в 1968 году. В 1970 году, первые результаты в этом направлении были получены *B.W. Boehm* [53, 54]. В 1981 году *B.W. Boehm* создал модель для оценки стоимости программного обеспечения *COnstructive COst MOdel* (*COCOMO*), а в 2000 году, усовершенствовав ее создал модель COCOMO II [55], которая широко применяется до настоящего времени.


Решение проблем экономичной разработки и сопровождения программного обеспечения привело в 2003 году к разработке бережливых технологий (lean software development), основанных на концепции повторного использования и бережливого подхода, используемого с 1962 года фирмой *Toyota* для производства своей автомобильной продукции [56]. Использование этого подхода в инженерии программного обеспечения было описано *M. Poppendieck* и *T. Poppendieck* [57].

Зеленые информационные технологии и программное обеспечение, экосистемы. В 1992 году в Рио-де-Жанейро состоялась всемирная конференция, посвященная концепции устойчивого развития [58]. На этой конференции был принят подход к развитию мира, суть которого формулировалась так «живем сейчас, но помним о тех, кто будет жить в будущем». Реализация подхода предусматривалась в трех аспектах – экономическом, экологическом и социальном. Особенно актуальным был признан экологический аспект. Учитывая данное, во многих индустриях начали появляться зеленые технологии и системы (*green technologies and systems*). В контексте концепции устойчивого развития программное обеспечение является его активом, вследствие того, что оно продукт производственной деятельности человека. Современная производственная деятельность все чаще вступает в противоречие с процессами, поддерживающими устойчивый круговорот в биосфере. В. Вернадский указывал на необходимость решения задачи перехода от стихийных взаимодействий человека и биосферы к сознательным, которые превращают биосферу в ноосферу и обеспечивают устойчивое развитие. Сейчас, эта задача вновь актуальна в контексте программы совместных действий в интересах устойчивого развития, а программное обеспечение играет важную роль в ее решении. Поэтому, не удивительно, что такие работы начали появляться в инфор-


матике и в частности, в инженерии программного обеспечения [59]. Учеными *S. Murugesan* и *S. Naumann* в 2008 году были выполнены работы по зеленым информационным технологиям, устойчивой информатике и устойчивой инженерии программного обеспечения (*sustainable software engineering*) [60]. Применение экологических исследований в индустрии программного обеспечения показывает, что их распространение идет на основе трех принципов и в трех основных направлениях. Общими экологическими принципами, которыми следует руководствоваться в стремлении достичь экологических результатов предлагаются следующие [61, 62]: эко-эффективность (eco-efficiency); эко-справедливость (eco-equity); эко-результативность (eco-effectives). Основ-

ные направления, это экологическое производство и использование программного обеспечения; ресурсосберегающее и безотходное производство программного обеспечения; экосистемы программного обеспечения и программное обеспечение как экосистема В 2013 году *M. Lungu* выполнил первые работы по экосистемам программного обеспечения, а в августе 2013 года состоялся первый рабочий семинар по этой теме, посвященный архитектуре экосистем программного обеспечения [63, 64]. *M. Lehman* в работах по эволюции программного обеспечения показал, что оно характеризуется следующим:

- изменением (развитием) – непременное свойство программного обеспечения, обусловленное наличием обратных связей и связанное с законами эволюции программ;
- наличием метасистемы, которая включает субъекты и продукты деятельности, процессы и организацию, содержит большое количество обратных связей, стабилизирующих внутренних механизмов, влияющих на процессы планирования, управления и повышения их эффективности; эффективное планирование и обслуживание программы требует понимания ее места в метасистеме, а также взаимодействий как между элементами, так и внутри них. При этом, программы, о которых идет речь, по классификации *M. Lehman* являются E – программами, а метасистема, – это их внешняя среда – реальный мир. На таком взгляде на программное обеспечение и строится понимание экосистем. В этом аспекте рассматривается два типа взаимодействий – внешние и внутренние. Внешние взаимодействия обусловлены наличием других экосистем. Например, очень часто программное обеспечение используется в составе или рядом с другим программным обеспечением, о существовании которого разработчик не мог знать. При этом эволюция такого программного обеспечения зависит от эволюции других приложений, а интерес представляют задачи создания моделей таких экосистем и моделей их эволюции. Внутренние взаимодействия обусловлены наличием в программном обеспечении клонов программ, программ-агентов, «обществ» программ [65], а также производителями, потребителями, различными регламентирующими организациями, которые в свою очередь могут рассматриваться как экосистемы. Возникают задачи исследования устройства такого программного обеспечения, принципов взаимодействия членов «обществ» программ и «обществ» между собой.

Образование в инженерии программного обеспечения

Первый образовательный стандарт *SWEBOK (Software Engineering Body of Knowledge)* по инженерии программного обеспечения был создан в 2000 году. На сегодня действует версия 2015 года [69]. Руководство для создания планов и программ *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* для подготовки бакалавров было разработано в 2001 году. Сейчас действует версия 2015 года [70]. Магистерские программы каждый университет создает свои.

В Украине до 2006 года специалисты в области инженерии программного обеспечения готовились в рамках бакалавраата Компьютерные науки. В 2000 году в Национальном авиационном университете была открыта первая в Украине кафедра Инженерии программного обеспечения и совместно с Ассоциацией Информационные технологии Украины, в которую входили предприятия индустрии программного обеспечения, началась подготовка к открытию бакалавраата по инженерии программного обеспечения [71–75]. В 2006 году такой бакалавраат был открыт, а осенью сделан первый набор студентов.

Учебный план был разработан на основе паттерна *N2S-1c* из *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* 2004 года. Сейчас большинство классических и политехнических университетов Украины готовят бакалавров по специальности 121 Инженерия программного обеспечения, а на базе специальности 01.05.03 Математическое и программное обеспечение вычислительных машин, комплексов и сетей создана докторантура специальности Инженерия программного обеспечения.

Выводы

Сейчас инженерия программного обеспечения, это одна из наибольших и прибыльных индустрий. Специальность инженера по программному обеспечению высоко востребована во всем мире среди абитуриентов, но умелых кадров попрежнему не хватает [10].

Вместе с тем наблюдается усиление интеграции Software engineering и System engineering, а также глобализации *Software-Intensive-Systems of Systems* и интероперабельности компонентов (возможность взаимодействия программных и аппаратных изделий разных поставщиков). В связи с этим повышается роль *COTS*, повторного использования и наследуемого программного обеспечения; бережливой инженерии программного обеспечения; экосистем программного обеспечения (экологии программного обеспечения); эмпирических и математических основ инженерии программного обеспечения.

Литература

1. Report on a conference sponsored by the NATO science committee, Garmisch, Germany, 7th to 11th October 1968, Editors: Peter Naur and Brian Randell.
2. Ершов А.П. Технология разработки систем программирования [Текст]. Системное и теоретическое программирование. Новосибирск. ВЦ СО АН СССР. 1972. С. 136–184.
3. Бондаренко М., Сидоров М., Морозова Т., Мендзебровський І. Модель випускника бакалаврату «Програмна інженерія», Вища школа. 2009. № 4. С. 50–61.
4. Boehm B. 2006, A View of 20th and 21st Century Software engineering [Text]. ICSE'06. May 20–28. China. 2006. P. 12–29.
5. Ершов А.П., Шура-Бура. Пути развития программирования в СССР [Текст]. Кибернетика. 6. 1976. С. 141–155.
6. Royce W. Managing the development of large software systems: Concepts and techniquies [Text]. Proc. Of WESCON. Aug. 1970.
7. Brooks F.P. The Mythical Man-Month: Essays on Software Engineering. [Text]. 1st ed. Addison–Wesley. 1975. P. 200.
8. Brooks F.P. No silver bullet: Essence and accidents of software engineering. [Text]. IEEE Computer. 20(4):10-19, April. 1987.
9. It Ukraine from a to z, http://www.uadn.net/files/ua_hightech.pdf
10. Розвиток української it-індустрії, Аналітичний звіт, It Ukraine, 2018.
11. https://dou.ua/lenta/articles/jobs-and-trends-2017/?from=doufp.
12. Ершов А.П. Научные основы доказательного программирования [Текст]. Вестн. АН СССР. 1984. № 10. С. 9–19.
13. Глушков В.М., Вельбицкий И.В. Технология программирования и проблемы ее автоматизации. [Текст]. *УСИМ*. Киев. № 6. 1976. С. 75–93.
14. Вельбицкий И.В. Технология программирования. [Текст]. Техника. Киев. Украина. 1984. 279 с.
15. International standart ISO/IEC 8631. Information technology-Program constructs and convention for their Representation – Second edition 1989.08.01 Geneve 20, Switzerland: ISO/IEC Copyright Office, P. 7. 1989.
16. Липаев В.В. Надежность программного обеспечения АСУ. [Текст]. М.: Энергоиздат, 1981.
17. Липаев В.В. Качество программного обеспечения. [Текст]. М.: Финансы и статистика, 1983.
18. Дал У., Дейкстра Э., Хоор К. Структурное программирование = Structured Programming. 1-е изд. М.: Мир, 1975. 247 с.

19. Dijkstra E.W. Go To Statement Considered Harmful. *Communications of the ACM*. Vol. 11. N 3, March 1968. P. 147–148.
20. Bohm Corrado; Giuseppe Jacopini (May 1966). "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". *Communications of the ACM*. **9** (5): 366–371. doi:10.1145/355592.365646
21. Wilkes, Maurice (1951). "The EDSAC Computer". *Proceedings of the Review of Electronic Digital Computers*: 79. doi:10.1109/AFIPS.1951.13
22. Бауэр Ф., Гооз Г. Информатика. М.: Мир, 1976. 484 с.
23. Segmentation and Design Strategies for Modular Programming." In T. O. Barnett and L. L. Constantine (eds.), Modular Programming: Proceedings of a National Symposium. Cambridge, Mass.: Information & Systems Press, 1968.
24. Parnas D.L. (December 1972). "On the Criteria To Be Used in Decomposing Systems into Modules". *Communications of the ACM*. **15** (12): 1053–58. doi:10.1145/361598.361623
25. Wirth N. Programming in Modula-2. Springer-Verlag, Heidelberg, New York, 1982.
26. Jean Ichbiah (October 1984). «Ada: Past, Present, Future – An Interview with Jean Ichbiah, the Principal Designer of Ada». *Communications of the ACM*. **27** (10). P. 990–997. doi:10.1145/358274.358278
27. Dahl O.-J., Myhrhaug B., Nygaard K. SIMULA67, Common base language/-Oslo. 1968. 96 p.
28. Hoare C. A. R. An axiomatic basis for computer programming, Comm. Of ACM, 12(1969). P. 576–580.
29. Wirth N., Weber H. EULER: A generalization of ALGOL, and its formal definition, Comm/ of ACM. 1966. 9. P. 13–25.
30. Goldberg A., Robson D. SmallTalk 80 The language and its implementation, Addison-Wesley, New-York, 1983.
31. Maier M.W. Architecting principels for systems-of-systems, Systems engineering, 1, 4(1998). P. 267–284.
32. Royce W.W. Managing the development of large software systems Proceedings of IEEE WESCON, 1970. 26. P. 328–388.
33. Boehm B.W. Spiral Model of software Development and Enhancement. Computer. 1988. May. P. 61–73.
34. Sydorov M.O. Software engineering: lecture curs. K.:NAU, 2007. 140 p.
35. Lehman M.M. "Programs, life cycles, and laws of software evolution", *Proceedings of the IEEE*, September 1980. P. 1060–1076.
36. Neighbors J.M. Software construction using components, Ph.D. Thesis, Dept. of Information and Computer Science, University of California, Irvin, 1981.
37. Neighbors J.M. "the draco approach to constructings of t ware from reusable components, "IEEE transactions on software engineering. September 1984. Vol. 10, N 5. P. 564–574.
38. Boehm B.W. Improving Software Productivity. Computer. 1987. Vol. 20, N 9. P. 43–57.
39. Prieto-Diaz R. and Freeman P. "Classifying Software for Reusability," IEEE Software. P. 6–16. January 1987.
40. Biggerstaff T.J. "An Assessment and Analysis of Software Reuse," In Advances in Computers 34, M. Yovits, Ed., Academic Press, New York, NY. 1992. P. 1–57.
41. Biggerstaff T.J. and Richter C. "Reusability Framework, Assessment, and Directions," IEEE Software 4, 1987. 2. P. 41–49.
42. Biggerstaff T.J. Design recovery for maintenance and reuse, Computer, july, 1989. P. 36–49.
43. Chikofsky E.J., CrossII J.H. Reverse engineering and design recovery: a taxonomy, IEEE Software, January, 1990. P. 13–17.
44. Chikofsky E.J Foreword, Comm. Of ACM. Vol. 37, N 5. 1994. P. 24.
45. Software evolution and feedback, Theory and practice, edited by N. H. Madhavji, Wiley. 2006. 570 p.
46. Fenton N., Pfleeger S., Glass R., Science and substance: A challenge to software engineers, IEEE Software, 1994. 11(4). P. 86–95.
47. Basili, Editorial, Empirical software engineering journal, 1996 1(2).
48. Soloway E., Ehrlich K., Empirical studies of programming knowledge, IEEE Transactions on software engineering. 1984. Vol. 10, N 5. P. 595–607.
49. L.L. Constantine The Peopleware Papers: Notes on the Human Side of Software. NJ: Prentice Hall. 2001.
50. Humphrey W.S. Characterizing the Software Process: A Maturity Framework (CMU/SEI-87-TR-11, ADA182895). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1987.
51. Humphrey, Watts S. Managing the Software Process. Reading, MA: Addison-Wesley, 1989.

52. Software Engineering Institute. CMMI A-Specification, Version 1.3, July 15, 1998.
53. Software engineering code of ethics and professional practice (Version 5.2) as recommended by the IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices https://www.ics.uci.edu/~redmiles/ics131-FQ03/week08ethics/IEEE-ACM-Ethics
54. Gotterbarn D., Miller K., Rogerson S., SOFTWARE ENGINEERING CODE OF ETHICS, Comm. Of ACM. 1997. Vol. 4, N 11. P. 110–118.
55. Barry Boehm Software Engineering Economics IEEE Transactions on Software Engineering. 1984. Vol. SE-10 (1). P. 4–21
56. Barry Boehm. Software Engineering Economics. Prentice-Hall 1981.
57. Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. *Software Cost Estimation with COCOMO II* (with CD-ROM). Englewood Cliffs, NJ:Prentice-Hall. 2000.
58. Shingo S. Study of Toyota production system, Productivity press, 1981.
59. Poppendieck, Mary. Implementing lean software development : from concept to cash. Mary Poppendieck, Tom Poppendieck. 2007.
60. Orsato R.J. Sustainability strategies, Business press. 2009. 243 p.
61. Chen A.J.W., Boudrean M.-C., Watson R.T. Information systems and ecological sustainability. Journal Of Systems and Information Technology. 2008. Vol. 10. N 3. P. 186–201.
62. Dyllick I., Hockerts K. Beyond the business case for corporate sustainability. – Busenes strategy and the Environment. Vol. 11. P. 130–141.
63. Chen W., Watson R.T. Information systems and ecological sustainability. Journal of systems and Information technology. 2008. Vol. 140. N 3. P. 186–201.
64. Harnessing Green IT: Principles and Practices, (with GR Gangadharan; Eds), Wiley and IEEE Computer Society Press. 2012.
65. Любимский Э.З. На пути к построению общества программ. Программирование. 2009. № 1. С. 4–10.
66. Stefan Naumann, Sustainability Informatics – A new Subfield of Applied Informatics? Environmental Informatics and Industrial Ecology, Shaker Verlag, Aachen 2008.
67. Mircea Lungu, Towards reverse engineering software ecosystems 2008 IEEE International Conference on Software Maintenance.
68. Lungu M. Reverse Engineering Software Ecosystems.PhD thesis, University of Lugano, October 2009.
69. SWEBOK Guide V3.0б 2014, IEEE Society.
70. https://www.acm.org/binaries/content/assets/education/se2014.pdf
71. Сидоров М.О. Кафедра інженерії програмного забезпечення. Компьютер-Клас! 2001. № 8. С. 17–19.
72. Сидоров Н.А. Инженерия программного обеспечения – дисциплина или бакалаврат? Матеріали міжнародної науково-практичної конференції «Розробка систем програмного забезпечення: виклики часу та роль інформаційному суспільстві». Київ, 27-28 січня 2005 р.
73. Сидоров Н.А. Инженерия ПО -дисциплина или бакалаврат? Корпоративные системы. 2005. № 2. С. 22–27.
74. Сидоров Н.А. Инженерия программного обеспечения – учебная дисциплина или подготовка бакалавра? *Управляющие системы и машины*. 2006. № 2. С. 25–34.
75. Сидоров Н.А., Медзебровский И.Б. Подготовка в Украине инженеров по программному обеспечению. Управление качеством инженерного образования и инновационные образовательные технологии: Сборник докладов Международной научно-методической конференции. Москва, 28–30 октября 2008. М.: МГТУ им. Н.Э. Баумана, 2008. Ч. 1. С. 14–18.

References

1. Report on a conference sponsored by the NATO science committee, Garmisch, Germany, 7th to 11th October 1968, Editors: Peter Naur and Brian Randell.
2. Ershov A.P. The programming systems developing technology, System and theoretical programming. 1972. P. 136–184.
3. Bondarenko M., Sydorov M., Morozova T., Mendzebrovskiy I. Model of a graduate of Bachelor's degree "Software engineering", Higher education. 2009. N 4. P. 50–61.
4. Boehm B., 2006, A View of 20th and 21st Century Software engineering [Text]. ICSE'06. May 20–28. China. 2006. P. 12–29.

5. Ershov A.P. The ways of programming developing in USSR, Cebernetic. 6. 1976. P. 141–155.
6. Royce W. Managing the development of large software systems: Concepts and techniquies [Text]. Proc. Of WESCON. Aug. 1970.
7. Brooks F.P. The Mythical Man-Month: Essays on Software Engineering. [Text]. 1st ed. Addison–Wesley. 1975. P. 200.
8. Brooks F.P. No silver bullet: Essence and accidents of software engineering. [Text]. IEEE Computer. 20(4):10-19, April. 1987.
9. It Ukraine from a to z, http://www.uadn.net/files/ua_hightech.pdf
10. The developing of Ukrainian it-industry, Analytical report. 2018.
11. https://dou.ua/lenta/articles/jobs-and-trends-2017/?from=doufp.
12. Ershov A.P. Science foundations of evidence programming, AS USSR proceeding. 1984. N 10. P. 9–19.
13. Glushkov V., Velbytsky I, Programming technology and its problems implementation, USIM. N 6. 1976. P. 75–93.
14. Velbytsky I. Programming technology, Technic. 1984. 279 p.
15. International standart ISO/IEC 8631. Information technology-Program constructs and convention for their Representation – Second edition 1989.08.01 Geneve 20, Switzerland: ISO/IEC Copyright Office, P. 7. 1989.
16. Lipaev V.V. ASM Software reliability. 1981.
17. Lipaev V.V. Software quality, Finances and statistic, 1983.
18. Dijkstra E.W. Structured Programming. 1975. 247 p.
19. Dijkstra E.W. Go To Statement Considered Harmful. *Communications of the ACM*. Vol. 11. N 3, March 1968. P. 147–148.
20. Bohm, Corrado; Giuseppe Jacopini (May 1966). "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules". *Communications of the ACM*. **9** (5): 366–371. doi:10.1145/355592.365646.
21. Wilkes, Maurice (1951). "The EDSAC Computer". *Proceedings of the Review of Electronic Digital Computers*: 79. doi:10.1109/AFIPS.1951.13
22. Bauer F., Gooz G. Informatics. Mir, M.: 1976. 484 p.
23. Segmentation and Design Strategies for Modular Programming." In T. O. Barnett and L. L. Constantine (eds.), Modular Programming: Proceedings of a National Symposium. Cambridge, Mass.: Information & Systems Press, 1968.
24. Parnas D.L. (December 1972). "On the Criteria To Be Used in Decomposing Systems into Modules". *Communications of the ACM*. **15** (12): 1053–58. doi:10.1145/361598.361623.
25. Wirth N. Programming in Modula-2. Springer-Verlag, Heidelberg, New York, 1982.
26. Jean Ichbiah (October 1984). «Ada: Past, Present, Future – An Interview with Jean Ichbiah, the Principal Designer of Ada». *Communications of the ACM*. **27** (10). P. 990–997. doi:10.1145/358274.358278.
27. Dahl O.-J., Myhrhaug B., Nygaard K. SIMULA67, Common base language/-Oslo. 1968. 96 p.
28. Hoare C.A.R. An axiomatic basis for computer programming, Comm. Of ACM, 12(1969). P. 576–580.
29. Wirth N., Weber H. EULER: A generalization of ALGOL, and its formal definition, Comm/ of ACM. 9. 1966. P. 13–25.
30. Goldberg A., Robson D. SmallTalk 80 The language and its implementation, Addison-Wesley, New-York, 1983.
31. Maier M.W. Architecting principels for systems-of-systems, Systems engineering, 1, 4(1998). P. 267–284.
32. Royce W.W. Managing the development of large software systems Proceedings of IEEE WESCON, 1970. 26. P. 328–388.
33. Boehm B.W. Spiral Model of software Development and Enhancement. Computer. 1988. May. P. 61–73.
34. Sydorov M.O. Software engineering: lecture curs. K.:NAU, 2007. 140 p.
35. Lehman M.M. "Programs, life cycles, and laws of software evolution", *Proceedings of the IEEE*, September 1980. P. 1060–1076.
36. Neighbors J.M. Software construction using components, Ph.D. Thesis, Dept. of Information and Computer Science, University of California, Irvin, 1981.
37. Neighbors J.M. "the draco approach to constructings of t ware from reusable components, "IEEE transactions on software engineering. September 1984. Vol. 10, N 5. P. 564–574.
38. Boehm B.W. Improving Software Productivity. Computer. 1987. Vol. 20, N 9. P. 43–57.
39. Prieto-Diaz R. and Freeman P. "Classifying Software for Reusability," IEEE Software. P. 6–16. January 1987.

40. Biggerstaff T.J. “An Assessment and Analysis of Software Reuse,” In Advances in Computers 34, M. Yovits, Ed., Academic Press, New York, NY. 1992. P. 1–57.
41. Biggerstaff T.J. and Richter C. “Reusability Framework, Assessment, and Directions,” IEEE Software 4, 1987. 2. P. 41–49.
42. Biggerstaff T.J. Design recovery for maintenance and reuse, Computer, july, 1989. P. 36–49.
43. Chikofsky E.J., CrossII J.H. Reverse engineering and design recovery: a taxonomy, IEEE Software, January, 1990. P. 13–17.
44. Chikofsky E.J Foreword, Comm. Of ACM. Vol. 37. N 5. 1994. P. 24.
45. Software evolution and feedback, Theory and practice, edited by N. H. Madhavji, Wiley. 2006. 570 p.
46. Fenton N., Pfleeger S., Glass R., Science and substance: A challenge to software engineers, IEEE Software, 1994. 11(4). P. 86–95.
47. Basili, Editorial, Empirical software engineering journal, 1996 1(2).
48. Soloway E., Ehrlich K., Empirical studies of programming knowledge, IEEE Transactions on software engineering. 1984. Vol. 10. N 5. P. 595–607.
49. L.L. Constantine The Peopleware Papers: Notes on the Human Side of Software. NJ: Prentice Hall., 2001.
50. Humphrey W.S. Characterizing the Software Process: A Maturity Framework (CMU/SEI-87-TR-11, ADA182895). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1987.
51. Humphrey, Watts S. Managing the Software Process. Reading, MA: Addison-Wesley, 1989.
52. Software Engineering Institute. CMMI A-Specification, Version 1.3, July 15, 1998.
53. Software engineering code of ethics and professional practice (Version 5.2) as recommended by the IEEE-CS/ACM Joint Task Force on Software Engineering Ethics and Professional Practices https://www.ics.uci.edu/~redmiles/ics131-FQ03/week08ethics/IEEE-ACM-Ethics
54. Gotterbarn D., Miller K., Rogerson S. SOFTWARE ENGINEERING CODE OF ETHICS, Comm. Of ACM. 1997. Vol. 4, N 11. P. 110–118.
55. Barry Boehm Software Engineering Economics IEEE Transactions on Software Engineering. 1984. Vol. SE-10 (1). P. 4–21
56. Barry Boehm. Software Engineering Economics. Prentice-Hall 1981.
57. Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. *Software Cost Estimation with COCOMO II* (with CD-ROM). Englewood Cliffs, NJ:Prentice-Hall. 2000.
58. Shingo S. Study of Toyota production system, Productivity press, 1981.
59. Poppendieck, Mary. Implementing lean software development : from concept to cash. Mary Poppendieck, Tom Poppendieck. 2007.
60. Orsato R.J. Sustainability strategies, Business press. 2009. 243 p.
61. Chen A.J.W., Boudrean M.-C., Watson R.T. Information systems and ecological sustainability. Journal Of Systems and Information Technology. 2008. Vol. 10. N 3. P. 186–201.
62. Dyllick I., Hockerts K. Beyond the business case for corporate sustainability. – Busenes strategy and the Environment. Vol. 11. P. 130–141.
63. Chen W., Watson R.T. Information systems and ecological sustainability. Journal of systems and Information technology. 2008. Vol. 140. N 3. P. 186–201.
64. Harnessing Green IT: Principles and Practices, (with GR Gangadharan; Eds), Wiley and IEEE Computer Society Press. 2012.
65. Lyubimsky E.Z. Towards a society building programs. Programming. N 1. 2009. P. 4–10.
66. Stefan Naumann, Sustainability Informatics – A new Subfield of Applied Informatics? Environmental Informatics and Industrial Ecology, Shaker Verlag, Aachen 2008.
67. Mircea Lungu, Towards reverse engineering software ecosystems 2008 IEEE International Conference on Software Maintenance.
68. Lungu M. Reverse Engineering Software Ecosystems.PhD thesis, University of Lugano, October 2009.
69. SWEBOK Guide V3.0б 2014, IEEE Society.
70. https://www.acm.org/binaries/content/assets/education/se2014.pdf
71. Sydorov M. Software engineering department, Computer-Class. 2001. N 8. P. 17–19.
72. Sydorov M. Is the software engineering subject or postgraduate, Proceeding of Conference “Software systems developing – the role in information society. January 27–28, 2005.
73. Sydorov M. Is the software engineering subject or postgraduate, Corporate Systems. N 2, 2005. P. 22–27.
74. Sydorov M. Is the software engineering education subject or postgraduate, USIM. 2006. N 2. P. 25–34.

75. Sydorov M., Menzebrovsky I. Software engineer education in Ukraine, Proceeding of Conference “Management of Quality of engineering education and education innovation technologies”. 2008. Part 1. P. 14–18.

Об авторе:

Сидоров Николай Александрович,
доктор технических наук,
профессор.
Количество научных публикаций в украинских изданиях – 118.
Количество научных публикаций в зарубежных изданиях – 12.
http://orcid.org/0000-0003-0800-1668.

Получено 31.10.2018

Место работы автора:

Межрегиональная академия
управления персоналом,
03039, Киев,
ул. Фрометовская, 2.
Моб. тел.: 067 7980361.
Тел.: 044 2343600.
E-mail: NykSydorov@gmail.com