DOI: 10.24411/1993-8314-2020-10004

***О. В. Саяпин**, докт. техн. наук, доцент,*
Академия гражданской защиты МЧС России,
г. Москва, o.saiapin@amchs.ru

***О. В. Тиханычев**, канд. техн. наук, Группа компаний «Техносерв»,*
г. Москва, tow65@yandex.ru

***С. В. Чискидов**, канд. техн. наук, доцент,*
Московский авиационный институт (Национальный исследовательский университет),
г. Москва, aleksankov.sergey@gmail.com

***И. А. Быстракова**, магистрант, Московский городской педагогический университет,*
г. Москва, bystrakovair@mail.ru

Разработка интерфейсов прикладных программ: макетирование или прототипирование

Разработка прикладных программ – сложный процесс, связанный с определёнными трудностями, в том числе с созданием эффективных пользовательских интерфейсов. Проведённый авторами анализ показал наличие ряда проблем в данной области, определяющихся тем, что она находится на стыке научных дисциплин: теории управления, эргономики, технической эстетики, психологии.

В результате анализа факторов, влияющих на эффективность разработки пользовательских интерфейсов, синтезированы предложения по решению проблемы, основанные на использовании средств стандартизации, унификации и прототипирования.

Анализ показал, что для условий разработки прикладного программного обеспечения наибольшей эффективностью обладают специализированные системы прототипирования интерфейсов. Предложено уточнить нормативную документацию, задающую разработку автоматизированных систем управления, для реализации в процессе их создания обязательного этапа прототипирования интерфейсов.

Ключевые слова: автоматизация управления, разработка прикладных программ, пользовательский интерфейс, макетирование и прототипирование, быстрое и эволюционное прототипирование

Введение

Несмотря на достаточно длительную практику разработки автоматизированных систем управления (АСУ), в данной области до настоящего времени имеется ряд нерешённых проблем. Одна из них – недостаточная оптимальность построения процесса разработки и использования программного обеспечения автоматизированных систем управления [1, 2, 3, 4]. Этот вопрос, в том числе, а может быть, в первую очередь, не решён из-за проблем организации разработки пользовательских интерфейсов прикладных программ, реализуемых в АСУ. В то же время взрывное развитие информатизации, внедрение «облачных» технологий, породивших тенденцию «гиперподключения», привели к тому, что компьютерные программы и системы активно используются во всех сферах деятельности, от крупных корпоратив-

ных систем управления типа ERP (Enterprise Resource Planning) до персональных планировщиков типа Wunderlist или ToRound. Все компоненты этих систем общаются с пользователем через интерфейс, и эффективность этого общения напрямую зависит от качества интерфейса. Ситуация обостряется по мере развития и практического внедрения информационных технологий во все сферы жизни человека: сложно убедить современное поколение digital native работать в программах с морально устаревшими интерфейсами. С учётом этих факторов проблема создания эффективных интерфейсов пользователя является крайне актуальной.

Некоторые проблемы создания пользовательских интерфейсов

Практика эксплуатации программного обеспечения показывает, что пользовательский интерфейс любой программы является важнейшим её компонентом, определяющим удобство использования и, как результат, желание человека её применять при формировании управленческих решений. Известно достаточно большое количество случаев, когда не слишком удобный интерфейс мешал внедрению потенциально хорошей программы [5, 6, 7].

Сложность задачи создания качественного интерфейса пользователя определяется тем, что данная проблема по своей сущности является комплексной, находящейся на стыке научных и практических дисциплин: математики, технической эстетики, психологии, эргономики. И, несмотря на достаточно долгую историю вопроса, рациональная организация разработки интерфейсов программ до настоящего времени не решена в полном объёме, проблема остаётся актуальной.

Как показывает опыт разработки программ, в том числе опыт лично авторов, возникновение указанной проблемы определяется целым рядом факторов объективной и субъективной природы:

- не всегда рациональная организация процесса разработки программной продукции, в первую очередь взаимодействия разработчика с конечным потребителем;
- слабое участие в разработке специалистов смежных областей, способных оценить потребности пользователя и сформулировать требования по пользовательским качествам интерфейса уже на подготовительном этапе;
- недостаточная унификация компонентов и функций пользовательских интерфейсов, разрабатываемых для программных продуктов схожего по функционалу или области применения назначения.

Совокупность этих факторов не позволяет обеспечить требуемое качество пользовательских интерфейсов разрабатываемых прикладных программ, что в итоге приводит к общему снижению эффективности автоматизированной поддержки принятия решений.

Существующие подходы к решению проблем разработки пользовательских интерфейсов

Практика показывает, что чаще всего для оптимизации разработки интерфейсов применяется наиболее простой и очевидный метод – унификация и стандартизация управляющих и отображающих элементов из их состава. Для обеспечения функциональной и визуальной унификации могут быть использованы различные подходы: организационные, технологические, алгоритмические.

В теории указанные подходы должны обеспечивать разработку высокоэффективных пользовательских интерфейсов. На практике это далеко не всегда реализуется. Для выявления сущности проблемы предлагается провести анализ особенностей подходов к разработке пользовательских интерфейсов прикладных программ.

Организационные подходы обеспечивают формирование общих требований к интерфейсам программ, их доведение до разработчиков и контроль исполнения. В нашей

стране эти требования установлены в ГОСТ серии 34 (для АСУ) и серии 19 (для отдельных программ), описываются в техническом задании на опытно-конструкторскую работу (ОКР) по разработке программного обеспечения и детализируются в форме руководящих указаний главного конструктора ОКР.

Указанные документы формируются с опорой на специализированные стандарты, созданные на основе международных аналогов и принятые к использованию в РФ:

- в части функциональных требований к отдельным компонентам двумерных графических интерфейсов WIMP (window, icon, menu, pointing device) – стандарты ISO 9241-12, ISO 9241-14, ISO 9241-16, ISO/IEC 10741, ISO/IEC 12581 и им подобные;
- в части эргономических характеристик – международные стандарты ISO 9241-10, ISO/IEC 13407, а также ГОСТ Р ИСО/МЭК 12119-2000, ГОСТ Р ИСО/МЭК 9126-93 и ряд других.

Зарубежные разработчики преимущественно пользуются комплексом стандартов по организации интерфейсов переносимых операционных систем (Portable operating system interfaces — POSIX). Входящие в POSIX стандарты принято разделять на четыре группы по степени детализации требований [8, 9, 10]:

- базовые, определяющие общие принципы построения, реализации и тестирования интерфейсов переносимых приложений – серии IEEE 1003.1, -2, -3, -4, -7;
- конкретизирующие требования к интерфейсам для операционных платформ – серии IEEE 1003.5, -9, -16, -19, -20;
- определяющие взаимодействие в распределённых открытых системах и телекоммуникационных сетях, а также защиту информации – IEEE 1003.8, -12, -15,-17,-6);
- регламентирующие процесс разработки программ – серия IEEE 1003.10, -11, -13, -14, -18.

Порядок применения этих стандартов регулируется Руководством POSIX Guide IEEE 1003.0. Руководитель каждого конкретного проекта на базе указанного набора стандартов формирует правила описания пользовательских интерфейсов, обязательные для исполнения всеми участниками процесса [11, 12, 13, 14].

Технологические подходы реализуются через специализированные программные средства – фреймворки (Framework), типовые библиотеки интерфейсных компонентов, такие как Linux Mint, UI-kit и им подобные. Таким образом обеспечивается выбор интерфейсных компонентов из готовых наборов, а также централизованное управление этими наборами. Как показывает опыт применения таких средств, они просто автоматизируют использование унифицированных компонентов, упрощая труд разработчиков, но не внося в него ничего принципиально нового в части создания эффективных интерфейсных форм.

Таким образом, можно отметить, что организационные и технологические подходы, ориентированные на унификацию интерфейсов, в целом упрощают разработку пользовательских интерфейсов в рамках каждого отдельного проекта. В то же время анализ практики создания программного обеспечения показывает, что подходы, основанные на унификации компонентов, обеспечивают лишь частичное решение проблемы создания эффективного пользовательского интерфейса.

Не так давно появился ещё один вариант технологического решения данной проблемы, не связанный с унификацией, – создание интерфейсов с возможностью адаптации пользователем (настраиваемые интерфейсы) [15, 16, 17].

Развитием настраиваемых интерфейсов можно считать *алгоритмический подход*, при котором программа отслеживает действия пользователя, как успешные, так и ошибочные, и настраивает перечень, размеры и местоположение выводимых на экран элементов пользовательского интерфейса с учётом частоты и результативности его действий. Сейчас алгоритмический подход довольно активно развивается, его элементы уже ре-

ализованы в некоторых коммерческих программных продуктах, например в офисных приложениях. Данный подход может быть признан одним из наиболее перспективных принципов повышения эргономичности пользовательских интерфейсов. Хотя пока при его использовании существуют определённые проблемы.

Во-первых, алгоритмический подход достаточно сложен и пока полностью не отработан технически.

Во-вторых, и это главное, даже самые совершенные адаптивные интерфейсы настраиваются в определённых границах, которые, как и их базовую структуру, требуется задавать на начальном этапе процесса разработки. Это сохраняет ранее сформулированные требования к процессу создания входных и выходных форм программ, не устраняя обозначенную в статье проблему по существу.

Стоит отметить, что для преодоления недостатков существующих методов разработки пользовательских интерфейсов в практике разработки АСУ используется подход, основанный на так называемом макетировании. Основан он на том, что в любой из известных методологий разработки программ: рациональное программирование RUP (Rational Unified Process), итеративно-инкрементальный метод OpenUP, технологии быстрой разработки приложений RAD (rapid application development), методология MSF (Microsoft Solutions Framework), методология разработки программ с сертифицируемым уровнем надёжности Cleanroom (Cleanroom Software Engineering), группа технологий гибкого программирования Agile – в том или ином виде присутствует этап апробации с использованием макетов [18, 19, 20, 21, 22]. Именно он является основой приведения функционала и интерфейсов прикладных программ в соответствие требованиям заказчика путём их итерационной доработки. Данный этап, хоть не обязательный, но утверждён ГОСТ по разработке программной продукции.

Методология макетирования, реализуемая на практике через этап апробации, подразумевает создание пробной версии программного продукта с реализацией базовой части функционала (так называемой ß-версии) и её проверки на пригодность с привлечением конечных пользователей. Для реализации принципа макетирования, как правило, используются штатные средства разработки программ.

В то же время, как показывает личный опыт авторов, в применении методологии макетирования имеются определённые проблемы, обусловленные тем, что в структуре технического задания на ОКР по разработке программной продукции описание интерфейсов программ нормативными и руководящими документами не предусмотрено. В результате первоначальный вариант интерфейса программист разрабатывает сам, в соответствии со своим видением проблемы, в надежде доработать интерфейс на последующих этапах работы. При средней продолжительности ОКР по разработке программной продукции в 2–3 года первая апробация, как правило, проводится не ранее чем через полгода-год. С учётом времени проведения самой апробации и срока обработки её результатов до программиста требования пользователя по первому уточнению интерфейсов доходят через 10–12 месяцев. Времени на доработку остаётся не так много. Это существенно снижает эффективность использования методологии разработки пользовательских интерфейсов на основе макетирования.

Таким образом, как показал анализ сложившейся ситуации, существующие подходы обеспечивают только частичную оптимизацию процесса разработки пользовательских интерфейсов и ограничены в применении либо по функционалу, либо по времени. Возникает вопрос: как выйти из сложившейся ситуации?

Прототипирование программ как метод совершенствования разработки пользовательских интерфейсов

Как показывает анализ мирового опыта разработки программной продукции, альтернативой макетированию может стать методология прототипирования, а конкретнее – быстрого прототипирования (Rapid Prototyping или RP-технологии). В рамках данной методологии предусматривается создание прототипов программ с минимальной функциональностью, но с полным набором входных и выходных форм, совместная работа с ними будущего пользователя и разработчика с целью оперативного уточнения требований к интерфейсам в итеративном режиме. В настоящее время указанный метод наиболее активно используется при создании веб-приложений [23, 24, 25]. В этом сегменте он показал достаточно высокую эффективность, с учётом чего можно сделать вывод о возможности его внедрения при разработке интерфейсов прикладных программ различной реализации и назначения.

Для быстрого прототипирования могут использоваться самые разные подходы и реализующие их программные средства: от простейших визуальных форм, описываемых в Power Point или Visio Professional, до специализированных систем, обеспечивающих не только визуализацию, но и имитацию реакции интерфейсных форм на действия пользователя и взаимодействие с плагинами, таких как Screen Architect или GUI Design Studio.

При анализе возможностей указанных средств [26, 27, 28] обращает на себя внимание существенное различие в сложности и функциональности систем прототипирования, а также то, что большинство из них, как отмечено ранее, используется для разработки веб-интерфейсов. Последнее вполне объяснимо в рамках тенденции [29, 30] активного развития UX/UI-дизайна (User Experience / User Interface).

Прошедшее не так давно, в рамках этой же тенденции, разделение программистов на специалистов по разработкам фронтэнд (frontend) и бекэнд (back-end) не решило проблему создания эффективных интерфейсов в полной мере. Фронтэнд-программисты так и остались в первую очередь программистами. Но это разделение дало толчок развитию средств прототипирования в рамках фронтэнд-технологий. Активное развитие используемых в этих технологиях фреймворков (Front end Frameworks) даёт разработчику широкий спектр динамично развивающегося инструментария: React, Angular, Vue, Svelte и других (в данном ряду системы выстроены по степени популярности, определяемой по данным сайта stateofjs.com за 2019 г.). Данный инструментарий очень различается по сложности и функционалу, и многие его компоненты с успехом могут использоваться для создания прототипов пользовательских интерфейсов. Особенно учитывая то, что в составе многих систем имеются встроенные препроцессоры (JS, CSS, HTML) и средства автоматизированного тестирования интерфейсов.

В любом случае все инструменты быстрого прототипирования имеют свои особенности, преимущества и недостатки. А благодаря широкому спектру возможностей, как показывает практика, каждое из указанных средств находит свою нишу.

При анализе особенностей средств прототипирования следует помнить, что сам этот подход тоже имеет градации: кроме уже упомянутого быстрого существует ещё и эволюционное прототипирование. Быстрое прототипирование основано на создании прототипов, технически не связанных с разрабатываемым программным продуктом. Эволюционное подразумевает создание прототипа, близкого не только по свойствам, но и по применяемым технологиям к разрабатываемой программе и на основе его поэтапного наращивания позволяющего обеспечить разработку реальной программы.

По своим характеристикам эволюционное прототипирование близко к макетированию. Соответственно, для реализации каждого из этих подходов используются разные средства прототипирования.

В целом использование того или иного подхода определяется конкретными условиями разработки. Но главное не применение конкретных средств, а сам факт использования метода прототипирования, который позволяет существенно сократить цикл взаимодействия пользователя с разработчиком. Как показал опыт работы авторов над разработкой АСУ, благодаря средствам прототипирования длительность цикла взаимодействия при разработке пользовательского интерфейса может быть сокращена до 3–5 месяцев. А в случае переноса работ по прототипированию на этап предпроектных исследований сокращение может стать ещё более существенным. И всё это при обеспечении приемлемой функциональности, позволяющей оценить не только визуальные, но и функциональные особенности разрабатываемых интерфейсов. За счёт этого при реализации функциональности, достаточно близкой к макетированию, обеспечивается сокращение времени «отклика» в 2–3 раза.

Заключение

Сравнительный анализ существующих и предлагаемого подхода к разработке пользовательских интерфейсов прикладных программ позволяет сделать ряд выводов.

Во-первых, быстрое прототипирование обладает преимуществами других методов, не повторяя их недостатков.

Во-вторых, быстрое прототипирования обеспечивает отладку интерфейсных форм программ до начала этапа апробации, что позволит на этапе β-версии отлаживать только их функционал и не тратить время на доработку входных и выходных форм. А сокращение времени проведения ОКР – это практически всегда сокращение затрат.

Более того, простота и сетевые возможности систем прототипирования позволят активно включить как самих пользователей, так и необходимых специалистов по различным областям знаний непосредственно в разработку интерфейсных форм, перейдя от применяемого сейчас итерационного подхода к непрерывному процессу работы.

В-третьих, тенденции развития АСУ показывают, что функционал существующих средств прототипирования, как и других средств, используемых сейчас при разработке интерфейсов, недостаточен. В обозримой перспективе, вполне вероятно, будут создаваться не только стандартные графические, но и 3D-интерфейсы, а также аудио- и нейро-интерфейсные компоненты программ. Соответственно, потребуются и средства их прототипирования.

И последнее. Для использования средств быстрого прототипирования требуется корректировка нормативных документов, регламентирующих ведение ОКР по разработке программной продукции АСУ, как в части введения этапа прототипирования, так и привлечения специалистов из смежных областей. Но эти меры не так сложны и затратны, как представляется, а их реализация обещает дать существенный прирост эффективности разработки прикладных программ.

Обобщая, можно сделать вывод, что использование методов быстрого прототипирования обеспечит оптимизацию цикла разработки пользовательских интерфейсов и в итоге повышение эффективности процесса разработки АСУ в целом.

Список литературы

1. *Емельянов А. А.* Технология создания компьютерных моделей для систем поддержки принятия решений // Прикладная информатика. 2006. № 1 (1). С. 121–135.
2. *Тиханычев О. В.* Теория и практика автоматизации поддержки принятия решений. – М.: Эдитус, 2018. – 76 с.
3. *Емельянов А. А., Шильникова О. В., Емельянова Н. З.* Моделирование процесса поддержки работоспособ-

ности развивающейся АСУ // Прикладная информатика. 2015. № 5 (10). С. 93–108.
4. *Корончик Д. Н.* Пользовательские интерфейсы интеллектуальных систем // Кибернетика и программирование. 2012. № 1. С. 16–22. DOI: 10.7256/2306-4196.2012.1.13861.
5. Improving Web Site Usability and Appeal. MSN Usability research. July, 2017. – 13 p.
6. *Сартасова Е.* Как не потерять лояльность клиентов из-за устаревшего и неудобного интерфейса сайта // Деловой мир. Практический онлайн-журнал. URL: https://delovoymir.biz/kak-ne-poteryat-loyalnost-klientov-iz-za-ustarevshego-interfeysa.html (дата обращения: 19.12.2018)
7. *Тиханычев О. В.* Субъективные аспекты применения математического моделирования военных действий в практике работы органов военного управления // Военная мысль. 2011. № 10. С. 49–53.
8. *Quarterman J.S., Wilhelm S.* Unix, Posix and open systems: The open standards puzzle. N.Y., Addison - Wesley Publishing; Company, 1993. – 416 p.
9. Open systems handbook: A guide to building open systems. Digital Equipment Corporation. USA, 1991. – 112 p.
10. *Липаев В. В.* Управление разработкой программных средств. Методы, стандарты, технология. – М.: Финансы и статистика, 1993. – 160 с.
11. *Арефьев Р. А., Зудилова Т. В.* SOA паттерн проектирования пользовательских интерфейсов для мультиплатформенных приложений // Программные системы и вычислительные методы. 2016. № 2. С. 201– 209. DOI: 10.7256/2305-6061.2016.2.18627.
12. *Gamma E.* Pattern languages of program design. Addison-Wesley Longman Publishing Co. 1997. Pp. 79–85.
13. Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, p. 18–23.
14. *Shahzad, S., Granitzer, M., Tochtermann K.*: Designing User Interfaces through Ontological User Model, Proceedings of the Fourth International Conference on Computer Sciences and Convergence Information Technology ICCIT 2009, Seoul, Korea, 24-26 November 2009, pp. 99–104.
15. *Gauch S., Chaffee J., & Pretschner A.* (2003). Ontology-based personalized search and browsing. Web Intelligence and Agent Systems, 1, pp. 219–234.
16. *McAvoy L. M., Chen L. & Donnelly M.* (2012, September). An ontologybased context management system for smart environments. UBICOMM 2012. The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies; Barcelona, Spain. 23–28 September 2012; pp. 18–23.
17. *Ковальчук С. В., Князьков К. В., Чуров Т. Н., Смирнов П. А., Бухановский А. В.* Организация человеко-компьютерного взаимодействия в средах компьютерного моделирования на базе облачной инфраструктуры // Прикладная информатика. 2012. № 5 (41). С. 89–102.
18. *Лукинова О. В.* Методологические аспекты управления жизненным циклом информационной системы на основе инструментов функциональной стандартизации // Программные продукты и системы. 2016. № 4. С. 27–35.
19. *Сныткин Т. И., Поляков А. Е., Руденко Г. А.* Системное проектирование автоматизированной системы военного назначения с использованием нотаций Archimate 2.1 // Динамика сложных систем – XXI век. 2018. Т. 12. № 2. С. 44–55.
20. *Тиханычев О. В., Макарцев Л. В., Гахов В. Р.* Рациональная организация процесса разработки прикладного программного обеспечения как предпосылка успешной автоматизации поддержки принятия решений // Программные продукты и системы. 2017. № 4. С. 706–710.
21. *Штрик А. А.* Технологии и инструментальные средства создания программного обеспечения: состояние и перспективы // Программные продукты и системы. 1991. № 2. С. 57–64.
22. *Вичугова А. А.* Этапы, методы и средства конфигурирования информационных систем // Прикладная информатика. 2015. № 3 (57). С. 88–99.
23. *Трегубов А. С.* Разработка адаптивных контекстозависимых интерфейсов с использованием онтологических моделей // Кибернетика и программирование. 2017. № 6. С.50–56. DOI: 10.7256/2306-4196.2017.6.24747.
24. *Кейно П. П., Силуянов А. В.* Разработка и внедрение интерпретатора декларативного языка моделирования web-интерфейсов на высоконагруженных системах // Прикладная информатика. 2015. № 1 (55). С. 55–70.
25. *Кольчугина Е. А., Заваровский К. В.* Применение методов генетического программирования к разработке web-интерфейсов // Прикладная информатика. 2012. № 5 (41). С. 64–74.
26. Инструменты быстрого прототипирования. – URL: https://habr.com/post/70001 (дата обращения: 13.12.2018).
27. Интерактивное прототипирование с GUI Machine. – URL: https:// habr.com/post/70001 (дата обращения: 13.12.2018).
28. *Тиханычев О. В.* Пользовательские интерфейсы в автоматизированных системах: проблемы разработки // Программные системы и вычислительные методы. 2019. № 2. С. 11–22. DOI: 10.7256/2454-0714.2019.2.28443.
29. How Human. Memory Works: Tips for UX Designers. UX Planet. – URL: https://uxplanet.org (дата обращения: 09.01.2019).

30. User Research. Empathy Is the Best UX Policy. UX Planet. – URL: https://uxplanet.org (дата обращения: 9.01.2019).

Reference

1. Emelyanov A. A. *Tekhnologiya sozdaniya komp'yuternykh modelej dlya sistem podderzhki prinyatiya reshenij* [The technology of creating computer models for system solutions]. *Prikladnaya informatika* [Journal of Applied Informatics], 2006, vol. 1, no. 1, pp.121–135 (in Russian).
2. Tikhanychev O. V. *Teoriya I practica avtomatizatsyi podderzhki prinyatiya resheny* [Theory and practice of decision support automation]. Moscow, Editus Publ., 2018, 76 p (in Russian).
3. Emelyanov A. A., Shil'nikova O. V., Emelyanova N. Z. Simulation of the process developing MIS and supporting its working capacity. *Prikladnaya informatika* [Journal of Applied Informatics], 2015, vol. 10, no. 5, pp. 93–108 (in Russian).
4. Koronchik D. N. *Pol'zovatel'skie interfeisy intellektual'nykh system* [User interfaces of intelligent systems.]. *Kibernetika i programmirovanie* [Cybernetics and programming], 2012, no. 1, pp. 16–22. DOI: 10.7256/2306-4196.2012.1.13861 (in Russian).
5. Improving Web Site Usability and Appeal. MSN Usability research. July, 2017. – 13 p.
6. Sartasova E. *Kak ne poteryat' loyal'nost' klientov iza ustarevshego i neudobnogo interfeisa saita. Delovoi mir. Prakticheskii onlain-zhurnal* [How not to lose customer loyalty due to outdated and inconvenient website interface. Business world. Practical online magazine.]. Available at: https://delovoymir.biz/kak-ne-poteryat-loyalnost-klientov-iz-za-ustarevshego-interfeysa.html (access date 19.12.2018).
7. Tikhanychev O.V. *Sub'ektivnye aspekty primeneniya matematicheskogo modelirovaniya voennykh deistvii v praktike raboty organov voennogo upravleniya* [Subjective aspects of the application of mathematical modeling of military operations in the practice of the military authorities]. Voennaya mysl' [Military Thougyt], 2011, no. 10, pp. 49–53 (in Russian).
8. Quarterman J.S., Wilhelm S. Unix, Posix and open systems: The open standards puzzle. N.Y., Addison - Wesley Publishing; Company, 1993. 416 p.
9. Open systems handbook: A guide to building open systems. Digital Equipment Corporation. USA, 1991.
10. Lipaev V. V. *Upravlenie razrabotkoi programmnykh sredstv. Metody, standarty, tekhnologiya* [Management of software development. Methods, standards, technology]. Moscow: Finansy i statistika Publ., 1993. 160 p. (in Russian).
11. Aref'ev R. A., Zudilova T. V. *SOA pattern proektirovaniya pol'zovatel'skikh interfeisov dlya mul'tiplatformennykh prilozhenii* [SOA pattern of user interface design for multiplatform applications]. *Programmnye sistemy i vychislitel'nye metody* [Software systems and computational methods], 2016, no. 2, pp. 201–209. DOI: 10.7256/2305-6061.2016.2.18627 (in Russian).
12. Gamma E. Pattern languages of program design. Addison-Wesley Longman Publishing Co., 1997, pp. 79–85.
13. Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, p. 18–23.
14. Shahzad, S., Granitzer, M., Tochtermann K.: Designing User Interfaces through Ontological User Model, Proceedings of the Fourth International Conference on Computer Sciences and Convergence Information Technology ICCIT 2009, Seoul, Korea, 24-26 November 2009, pp. 99–104.
15. Gauch S., Chaffee J. & Pretschner A. (2003). Ontology-based personalized search and browsing. Web Intelligence and Agent Systems, 1, p. 219–234.
16. McAvoy L. M., Chen L. & Donnelly M. (2012, September). An ontologybased context management system for smart environments. UBICOMM 2012. The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies; Barcelona, Spain. 23–28 September 2012; pp. 18–23.
17. Koval'chuk S. V., Knyaz'kov K. V., Churov T. N., Smirnov P. A., Bukhanovskii A. V. *Organizatsiya cheloveko-komp'yuternogo vzaimodeistviya v sredakh komp'yuternogo modelirovaniya na baze oblachnoi infrastruktury* [Organization of human-computer interaction in computer simulation environments based on cloud infrastructure]. *Prikladnaya informatika* [Journal of Applied Informatics], 2012, no. 5 (41), pp. 89–102 (in Russian).
18. Lukinova O. V. Methodological Aspects of Information System Life Cycle Management Based on Functional standardization Tools. *Programmnye produkty i sistemy* [Software & Systems], 2016, no. 4, pp. 27–35 (in Russian).
19. Snytkin T. I., Polyakov A. E., Rudenko G. A. *Sistemnoe proektirovanie avtomatizirovannoj sistemy voennogo naznacheniya s ispol'zovaniem notatsij Archimate 2.1* [System Design of an Automated Military System Using Notation Archimate 2.1]. *Dinamika slozhnykh sistem – XXI vek* [Dynamics of complex systems - XXI century], 2018, no. 2 (12), pp. 44–55 (in Russian).
20. Tikhanychev O. V., Makartsev L. V., Gakhov V. R. Rational organization of an application software development process for decision support successful automation. *Programmnye produkty i sistemy* [Software & Systems], 2017, no. 4 (30), pp. 706–710 (in Russian).

21. Shtrik A. A. *Tekhnologii i instrumental'nye sredstva sozdaniya programmnogo obespecheniya: sostoyanie i perspektivy* [Technologies and tools for creating software: status and prospects]. *Programmnye produkty i sistemy* [Software products and systems], 1991, no. 2, pp. 57–64 (in Russian).
22. Vichugova A. A. *Etapy, metody i sredstva konfigurirovaniya informatsionnykh sistem* [Stages, methods and means of configuring information systems]. *Prikladnaya informatika* [Journal of Applied Informatics], 2015, no. 3 (57), pp. 88–99 (in Russian).
23. Tregubov A. S. *Razrabotka adaptivnykh kontekstozavisimykh interfeisov s ispol'zovaniem ontologicheskikh modelei* [Development of adaptive context-dependent interfaces using ontological models]. *Kibernetika i programmirovanie* [Cybernetics and programming], 2017, no. 6, pp. 50–56. DOI: 10.7256/2306-4196.2017.6.24747 (in Russian).
24. Keino P. P., Siluyanov A. V. *Razrabotka i vnedrenie interpretatora deklarativnogo yazyka modelirovaniya web-interfeisov na vysokonagruzhennykh sistemakh* [Development and implementation of a declarative modeling language interpreter for web interfaces on high-load systems]. *Prikladnaya informatika* [Journal of Applied Informatics], 2015, no. 1 (55), pp. 55–70 (in Russian).
25. Kolchugina E. A., Zavarovskii K. V. *Primenenie metodov geneticheskogo programmirovaniya k razrabotke web-interfeisov* [Application of genetic programming techniques to the development of web interfaces]. *Prikladnaya informatika* [Journal of Applied Informatics], 2012, no. 5 (41), pp. 64–74 (in Russian).
26. *Instrumenty bystrogo prototipirovaniya* [Rapid Prototyping Tools]. Habr. Available at: https://habr.com/post/70001 (access date: 13.12.2018).
27. Tikhanychev O. V. User interfaces in automated systems: design problems. Software systems and computational methods, 2019, no. 2, pp. 11–22. DOI: 10.7256/2454-0714.2019.2.28443.
28. Interaktivnoe prototipirovanie s GUI Machine [Interactive prototyping with GUI Machine]. Habr. Available at: https://habr.com/post/70001 (access date: 13.12.2018).
29. How Human. Memory Works: Tips for UX Designers. UX Planet. Available at: https://uxplanet.org (access date: 9.01.2019).
30. User Research. Empathy Is the Best UX Policy. UX Planet. Available at: https://uxplanet.org (access date: 9.01.2019).

DOI: 10.24411/1993-8314-2020-10004

O. Sayapin, Academy of Civil Protection, Ministry of Emergencies, Moscow, Russia, o.saiapin@amchs.ru
O. Tikhanychev, Company group «Technoserv», Moscow, Russia, tow65@yandex.ru
S. Chiskidov, Moscow Aviation Institute (National Research University), Moscow, Russia, aleksankov.sergey@gmail.com
I. Bystrakova, undergraduate of Moscow City Pedagogical University, Moscow, Russia, bystrakovair@mail.ru

Development of application program interfaces: layout or prototype

A recognized direction of improving the efficiency of the use of organizational and technical systems is management automation, which provides increased efficiency and reasonableness of decisions made. A significant role in increasing the efficiency of any automated system is played by its software. First, this thesis refers to the application or special software. Development of application programs is fraught with certain difficulties, including those associated with the efficiency of user interfaces created in its process. The analysis carried out by the authors showed that there are a number of problems in this field, which are determined by the fact that it is located at the intersection of scientific disciplines: control theory, ergonomics, technical aesthetics, and psychology. The article analyzes the factors affecting the effectiveness of the development of user interfaces. Based on the analysis, proposals for solving problems based on the use of standardization, unification and prototyping tools were synthesized. Prototyping methods are divided into evolutionary and rapid. The analysis showed that for the conditions of developing application software of automated decision support systems, the latter approach provides the greatest efficiency, namely, the use of specialized prototyping systems. The article proposes to refine the regulatory documentation, which defines the development of automated control systems, to be implemented in the structure of the process of creating such systems, an obligatory stage of interface prototyping

Keywords: management automation, application development, user interface, prototyping and prototyping, rapid and evolutionary prototyping

About authors:

O. Sayapin, *Dr of Engineering, Associate Professor*

O. Tikhanychev, *PhD in Engineering*

S. Chiskidov, *PhD in Engineering, Associate Professor*

I. Bystrakova, *Master's Student*

For citation: Sayapin O., Tikhanychev O., Chiskidov S., Bystrakova I. Development of application program interfaces: layout or prototype. *Prikladnaya informatika* – Journal of Applied Informatics, 2020, vol. 15, no. 1, pp. 47 - 56 (in Russian). DOI: 10.24411/1993-8314-2020-10004