МЕТАЛЛУРГИЯ

УДК 669.
DOI: 10.21122/1683-6065-2019-2-67-71

Поступила 15.05.2019
Received 15.05.2019

ИСПОЛЬЗОВАНИЕ МЕХАНИЗМОВ ООП ДЛЯ УНИФИКАЦИИ ПРОЦЕДУР РАБОТЫ С МЕСЯЧНЫМ ПЛАНОМ ПРОИЗВОДСТВА И ОТГРУЗКИ ОСНОВНОГО ПРОИЗВОДСТВА

Н. П. ПАПКОВ, Е. С. ОНЕГИНА, ОАО «БМЗ – управляющая компания холдинга «БМК», г. Жлобин, Гомельская область, Беларусь, ул. Промышленная, 37. E-mail: nativi3dd@mail.ru.

Описан вариант модернизации программного обеспечения (ПО), позволяющего сформировать месячные планы производства и отгрузки цехов основного производства, в рамках которого использовали некоторые механизмы объектно-ориентированного программирования.

Результатом работы стали унификация одинаковых действий при формировании месячных планов для различных цехов в рамках данного ПО, использование одних и тех же объектов в различных частях ПО с быстрой адаптацией под конкретные требования в данном месте, возможность использования полученных объектов во вновь разрабатываемом обеспечении различными подразделениями отделов по разработке.

Ключевые слова. *Планирование, месячный план, отгрузка, производство, унификация, объектно-ориентированное программирование, ABAP, наследование, инкапсуляция.*

Для цитирования. *Папков, Н. П. Использование механизмов ООП для унификации процедур работы с месячным планом производства и отгрузки основного производства / Н. П. Папков, Е. С. Онегина // Литье и металлургия. 2019. № 2. С. 67–71. DOI: 10.21122/1683-6065-2019-2-67-71.*

USING MECHANISMS OF OBJECT–ORIENTED PROGRAMMING FOR THE UNIFICATION OF PROCEDURES OF WORK WITH THE MONTHLY PLAN AND FOR MANUFACTURE AND SHIPMENT OF THE MAIN PRODUCTION

N. P. PAPKOV, E. S. ONEGINA, OJSC «BSW – Management Company of Holding «BMC», Zhlobin city, Gomel Region, Belarus, 37, Promyshlennya str. E-mail: nativi3dd@mail.ru

This article describes a version of the modernization of the software (SOFTWARE), which allows to create monthly production plans and shipping program for shops of the main production, which used some of the mechanisms of object-oriented programming.

The result of this work was the unification of the same actions in the formation of monthly plans for different shops within the framework of this SOFTWARE, the use of the same objects in different parts of the SOFTWARE with rapid adaptation to specific requirements in this place, the possibility of using the obtained objects in the newly developed provision by various departments of development.

Keywords. *Planning, monthly plan, shipping, production, unification, object-oriented programming, ABAP, inheritance, encapsulation.*

For citation. *Papkov N. P., Onegina E. S. Using mechanisms of object-oriented programming for the unification of procedures of work with the monthly plan and for manufacture and shipment of the main production. Foundry production and metallurgy, 2019, no. 2, pp. 67–71. DOI: 10.21122/1683-6065-2019-2-67-71.*

Месячное планирование производства и отгрузки

Для создания месячного плана производства и отгрузки выполняются следующие общие шаги:

Создание плана отгрузки

На основе данных о заключенных и планируемых контрактах управлением по сбыту продукции формируется план отгрузки готовой продукции на месяц на дальнее и ближнее зарубежье с указанием объемов и цен, а также транспортных расходов.

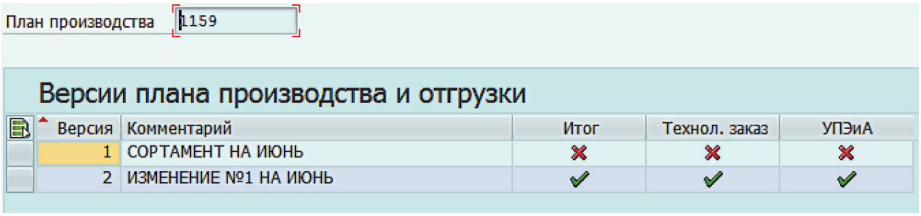

Рис. 1. Версии плана производства и отгрузки для стана 150 на июнь 2017 г.

Создание плана производства и отгрузки

Производственным управлением на основе плана отгрузки формируется план производства и отгрузки, в котором рассчитываются объемы производства.

Цены на отгружаемую продукцию и заявки на ВПС

На созданный план производства и отгрузки формируются цены на отгружаемую продукцию и создаются заявки на сырье и материалы, а также на вспомогательные производственные средства.

Месячное планирование производства и отгрузки для электросталеплавильных, прокатного и метизных цехов осуществляется в рамках системы SAP R/3 с помощью специально разработанных программ планирования, которые реализованы на языке ABAP. Каждая ABAP-программа по работе с месячным планированием производства или отгрузки позволяет проводить разные манипуляции с месячным планом: создавать его; изменять; просматривать; выгружать в локальную систему цеха; утверждать.

Для всех этих манипуляций существуют процедуры, которые схожи между собой по алгоритму, например, считывание версии плана производства или отгрузки цеха для определенного периода:

установить месяц, на который нужно считать версию плана;

считать заголовки версий плана на указанный месяц;

показать на экране список версий плана (рис. 1).

Каждый раз код полностью приходилось прописывать в программах, хотя отличались только входные параметры, например, такие, как «цех» или «дата».

Также в случае обнаружения ошибки в одной из программ планирования приходилось ее исправлять не только в целевой программе, но и во всех остальных программах, в которых использовались подобные процедуры. При этом большую роль в исправлении играл так называемый «человеческий» фактор, т. е. внес ли разработчик во все места нужные исправления или нет. Кроме того, если программу исправлял не автор, то вполне могла получиться ситуация, что ошибка исправлена только в одной программе, а не везде, где это требуется.

Поэтому возникал логичный вопрос: «*А почему бы не написать универсальный код для работы с месячным планом производства и отгрузки?*»

Необходимо было модернизировать существующие программы, что и повлияло на создание данной работы.

ABAP Objects

ABAP – внутренний язык программирования, который сочетает в себе объектно-ориентированные и процедурные элементы. В объектно-ориентированной части были адаптированы только те объектно-ориентированные концепции, которые доказали свою ценность для разработки приложений для предприятий в других языках.

Основные отличия ABAP Objects от других объектно-ориентированных языков связаны со средой разработки. В ABAP Objects можно использовать полный набор функций инструментальных средств ABAP.

Наследование и инкапсуляция ООП

Наследование – один из самых важных механизмов в ООП. Он позволяет создавать производные классы (классы наследники), взяв за основу все методы и элементы базового класса (класса родителя). Таким образом, экономится масса времени на написание и отладку кода новой программы. Объекты производного класса свободно могут использовать все, что создано и отлажено в базовом классе. При этом мы можем в производный класс дописать необходимый код для усовершенствования программы: добавить новые элементы, методы и т. д. Базовый класс останется нетронутым [1].

Классы в ABAP могут непосредственно наследоваться только от одного базового класса, хотя у базового класса может быть собственный базовый класс и т. д. Механизм наследования позволяет расширять или создавать конкретные классы от одного и более общего базового класса.

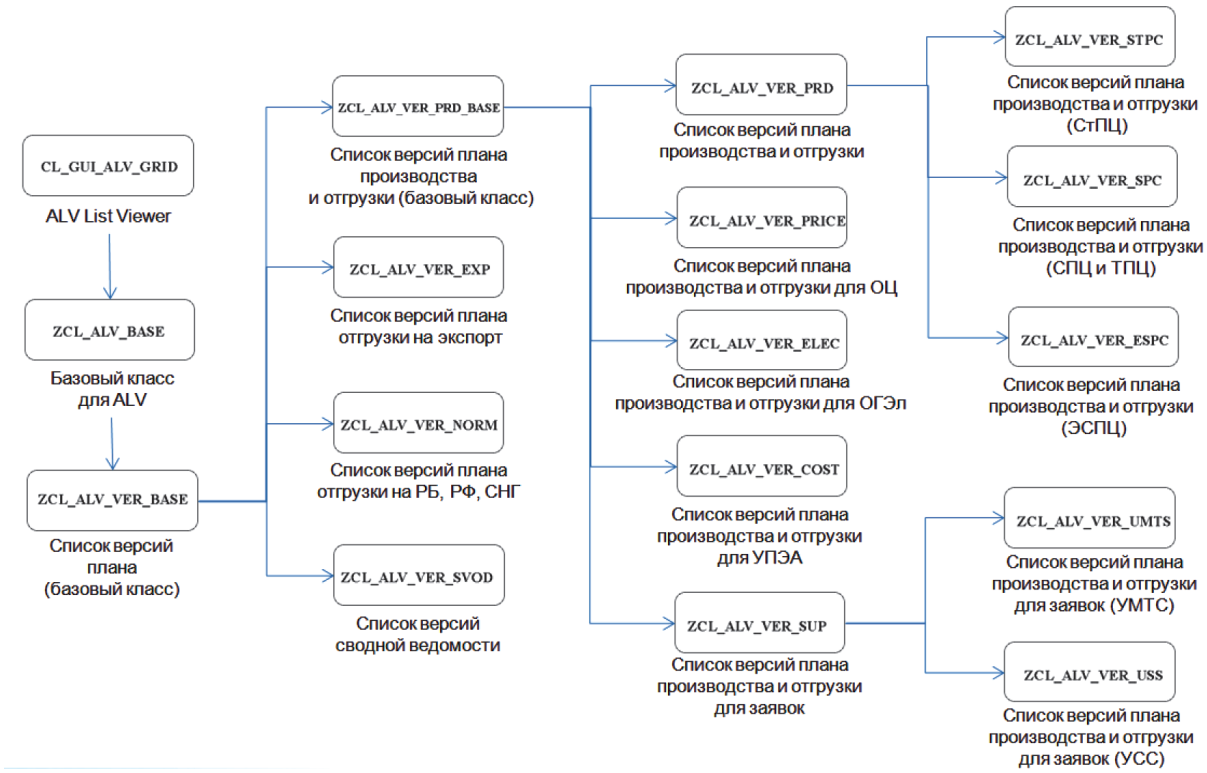

Рис. 2. Цепочка наследования классов

В качестве примера можно привести класс столов и его двух подклассов – кухонных и письменных столов. Все столы независимо от своего типа имеют длину, ширину и высоту. Пусть для письменных столов важна площадь поверхности, а для кухонных – количество посадочных мест. Общее выносится в класс, частное – в подклассы [2].

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.

Например, представьте себе автомобиль. Для того чтобы им управлять, Вам вовсе не нужно знать устройство двигателя внутреннего сгорания и всех систем внутри него [3]. Достаточно крутить руль и нажимать на педаль газа, не задумываясь, что в это время происходит с инжектором, дроссельной заслонкой и распредвалом. Именно сокрытие внутренних процессов, происходящих в автомобиле, позволяет эффективно его использовать даже тем, кто не является профессионалом-автомехаником с 20-летним стажем. Это сокрытие в ООП и носит название инкапсуляции [4].

Реализация наследования

Проанализировав программы планирования, было решено применять **наследование** объектно-ориентированного программирования. В итоге была выстроена «цепочка» классов (рис. 2): коды, которые были абсолютно схожи для всех цехов, были вставлены в методы вышестоящих классов. Если были отличия, то метод переопределялся в нижестоящих классах для конкретного цеха.

Ранее на основе базового класса ALV системы SAP «CL_GUI_ALV_GRID» был унаследован класс «ZCL_ALV_BASE» с добавлением неких усовершенствованных методов, использующихся для работы с ALV.

Для осуществления нашей поставленной цели от этого класса был унаследован класс «ZCL_ALV_VER_BASE» (рис. 3), в который были добавлены методы, работающие с месячными планами цехов: задание периода, считывание версий плана цеха за определенный период, их утверждение, а также, например, метод с общим для всех цехов описанием структуры плана.

Ниже по цепочке от этого класса было создано несколько дочерних классов:

1. Два из них позволяют считывать месячные планы отгрузки: «ZCL_ALV_VER_NORM» – список версий плана отгрузки на РБ, РФ, СНГ и «ZCL_ALV_VER_EXP» – список версий плана отгрузки на экспорт.

2. Третий класс «ZCL_ALV_VER_PRD_BASE» – базовый класс для формирования списка версий плана производства и отгрузки, на основании которого был создан класс «ZCL_ALV_VER_PRD». От

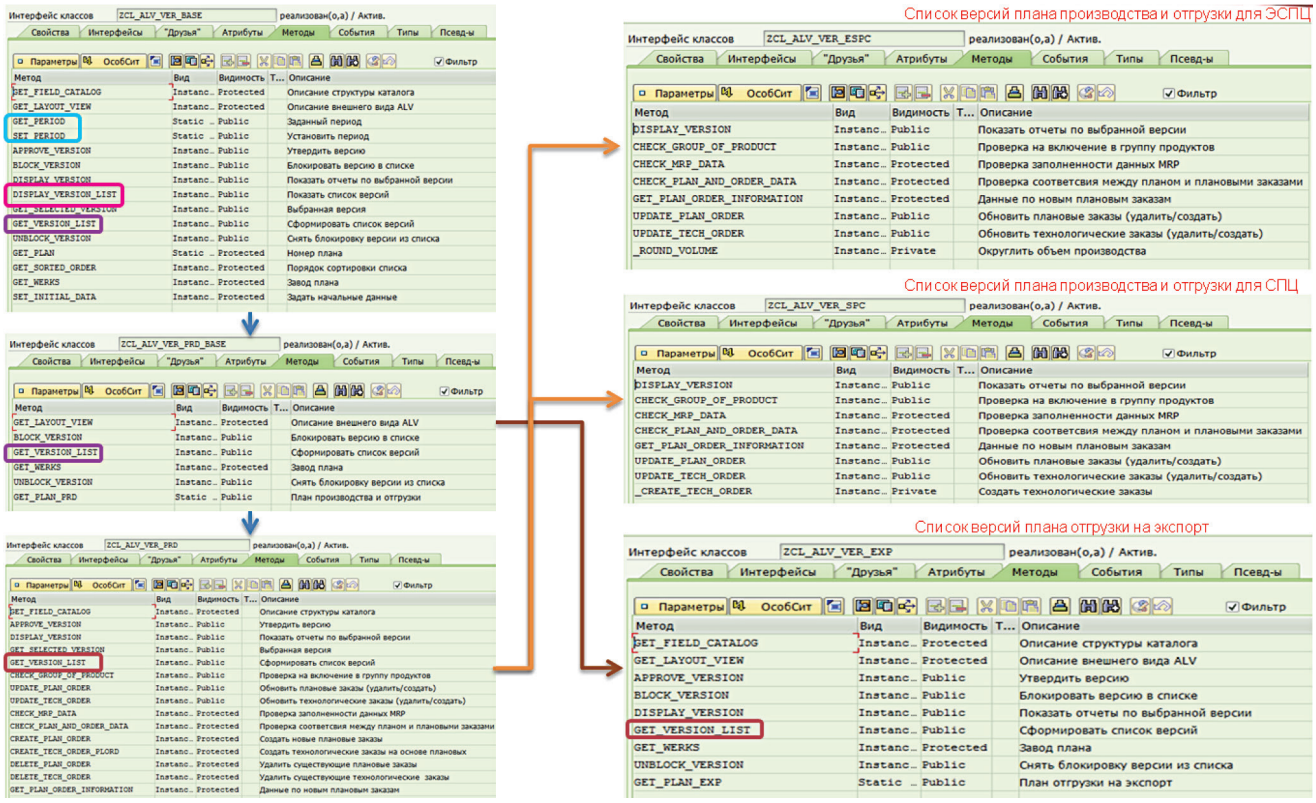

Рис. 3. Реализованные классы и методы

класса «ZCL_ALV_VER_PRD» пошло разъединение по производству. Было унаследовано три класса: для сталеплавильного – «ZCL_ALV_VER_ESPC», прокатного – «ZCL_ALV_VER_SPC» и метизного производств – «ZCL_ALV_VER_STPC» со своими либо переопределенными, либо добавленными для конкретного цеха методами.

Например, процедура установки периода, за который нужно считать план, не зависела от выбора цеха, поэтому она смело была вынесена в базовый класс «Список версий плана» set_period (get_period). Теперь, для того чтобы установить период, необходимо:

Инициализировать объекты

go_sdex_vers TYPE REF TO zcl_alv_ver_exp – для плана отгрузки на дальнее зарубежье,

go_sdno_vers TYPE REF TO zcl_alv_ver_norm – для плана отгрузки на ближнее зарубежье,

go_version TYPE REF TO zcl_alv_ver_espc – для плана производства и отгрузки ЭСПЦ,

**go_version TYPE REF TO zcl_alv_ver_spc* – для плана производства и отгрузки СПЦ,

**go_version TYPE REF TO zcl_alv_ver_stpc* – для плана производства и отгрузки СтПЦ.

Вызвать метод (одинаково во всех программах)

go_sdex_vers ->set_period() – для плана отгрузки на дальнее зарубежье,

go_sdno_vers ->set_period() – для плана отгрузки на ближнее зарубежье,

go_version ->set_period() – для плана производства и отгрузки.

При вызове метода *set_period()* все три объекта соответствующих им классов обращаются к одному и тому же методу *set_period()* класса «*ZCL_ALV_VER_BASE*».

Таким образов, наследование и переопределение позволили выстроить иерархию классов, вынося общие по алгоритму действия на уровень ближе к корню иерархии, а то, что отличалось – к ее листьям. При этом широко использовался в рамках иерархии вызов реализаций вышестоящих классов (классов-предков) с корректировкой при необходимости в нижестоящих классах (классах-потомках). Такой подход позволил настроить выполнение каждого действия для каждого плана производства и отгрузки с учетом тех особенностей, которые возникают в процессе работы с ним.

Инкапсуляция позволила ограничить доступ к данным объектам и способам их работы, т. е. каждый объект представлял строго определенный набор методов для работы с данными, убирая всю черновую работу с ними в себя. Это позволило в итоге определить тот набор методов, который может использоваться в любой программе планирования.

Итоги по модернизации

Месячные программы планирования были модернизированы с учетом создания «цепочки наследования». Они продолжают эксплуатироваться работниками производственного управления и управления по сбыту продукции. В процессе работы классы для работы со списком версий месячного плана производства и отгрузки дорабатываются.

В итоге применение наследования ООП уменьшило программный код при написании программ, ведь раньше каждую подпрограмму приходилось, можно сказать, дублировать, заменяя, например, только название цеха. Также, если меняется наименование переменной, то уже не нужно это отслеживать в тексте программы, так как изменение в методе класса распространяется на все программы, использующие данный метод.

Кроме того, каждый класс может быть использован для получения списка версий плана производства и отгрузки в других программах с минимальными затратами.

ЛИТЕРАТУРА

1. **Marienko L.** Наследование классов [Электронный ресурс]. Режим доступа: http://cppstudio.com/post/10103/. – Дата доступа: 20.05.2017

2. **Наследование** [Электронный ресурс] – Режим доступа: https://younglinux.info/oopython/inheritance.php. – Дата доступа: 20.05.2017

3. **ASTRAFOX** Фасад (Facade) [Электронный ресурс]/ ASTRAFOX – Режим доступа: https://abap-blog.ru/oop/fasad-facade/. – Дата доступа: 20.05.2017

4. **Пайсон М.** ООП с примерами (часть 2) [Электронный ресурс] / М. Пайсон – Режим доступа: https://habr.com/post/87205/. – Дата доступа: 20.05.2017.

REFERENCES

1. http:// cppstudio.com/post/10103/.

2. https:// younglinux.info/oopython/inheritance.php.

3. https:// abap-blog.ru/oop/fasad-facade.

4. https:// habr.com/post/87205.