

ԿԱՌԱՎԱՐՄԱՆ ԱՎՏՈՄԱՏԱՑՎԱԾ ՀԱՄԱԿԱՐԳԱՅԻՆ ԿԱՐԳԱԲԵՐՄԱՆ ՈՐՈՇ ՀԱՐՑԵՐ

Հ. Ա. ՄԵԼԻՔՅԱՆ

*Տեխնիկական գիտությունների թեկնածու, դոցենտ, ԳՊՀ պրոֆեսոր,
ԳՊՀ ինֆորմատիկայի և ֆիզիկամաթեմատիկական
գիտությունների ամբիոնի վարիչ*

Որևէ առարկայական տիրույթի համար ստեղծվող կառավարման ավտոմատացված համակարգի (ԿԱՀ) կամ ավտոմատացված տեղեկատվական համակարգի (ԱՏՀ) ծրագրային ապահովումը (ԾԱ) կամ ինչպես երբեմն անվանում են՝ ծրագրային միջոցը ծրագրերից, ընթացակարգերից, կանոններից, ինչպես նաև, եթե նախատեսված է, դրանց ուղեկցող փաստաթղթերից և տվյալներից կազմված ամբողջություն է, որով իրականացվում են տեղեկատվության մշակման համակարգի հաշվողական գործընթացները: ԿԱՀ-ի ծրագրային ապահովումը բաժանվում է երկու հիմնական մասի.

- ընդհանուր կամ ներքին ԾԱ,
- հատուկ կամ կիրառական ԾԱ:

Ընդհանուր ծրագրային ապահովումը նախատեսված է հաշվողական գործընթացների կազմակերպման, ինչպես նաև ծրագրային ապահովման ստեղծման և ստուգման տեխնոլոգիական գործընթացների ավտոմատացման համար:

Հաշվողական գործընթացների կազմակերպման ծրագրային ապահովումը (օպերացիոն համակարգը) իրականացնում է տեղեկատվության մուտք - ելքի կազմակերպման, ընդհատումներին արձագանքման, հաշվողական պրոցեսների և տեխնիկական միջոցների ֆունկցիոնալ վերահսկման, խնդիրների և հանձնարարությունների գերակայության սկզբունքով կատարման, տեղեկատվության վերականգնման և ավտոմատացման միջոցների փոխդասավորության փոփոխության և այլ գործառույթներ:

Ծրագրային ապահովման ստեղծման և ստուգման տեխնոլոգիական գործընթացների ավտոմատացման համալիրը բաղկացած է ծրագրավորման ապահովման «ծրագրերի կարգաբերման» փաստաթղթերի թողարկման և

շտկման, ծրագրային ապահովման մշակման ընթացքի ստուգման և այլ միջոցներից:

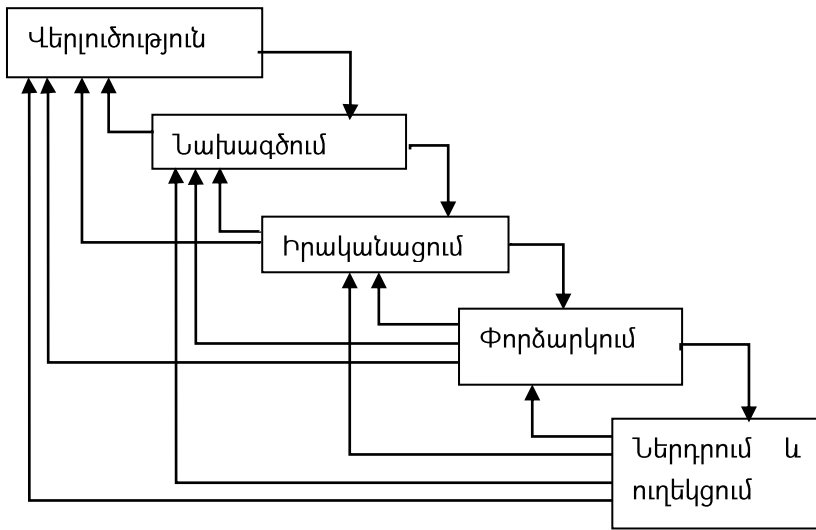
Հատուկ կամ կիրառական ծրագրային ապահովումը (ԿԾԱ) ծառայում է ԿԱՀ-ի տեղեկատվության հավաքման, մշակման և արտապատկերման կոնկրետ ֆունկցիոնալ խնդիրների իրականացման համար:

ԿԱՀ-ի ընդհանուր ԾԱ-ն հիմնականում հավաստագրված ծրագրային պատրաստի փաթեթներ են, որոնք ձեռք են բերվում և տեղադրվում ԿԱՀ-ի օբյեկտներում տեխնիկական միջոցների հետ միասին: Հետևաբար ԾԱ-ի ստեղծման աշխատանքների հիմնական մասը կիրառական ծրագրային ապահովման մշակումն է, որը բավականաչափ մեծ համակարգերի համար երկարատև և աշխատատար գործընթաց է: ԿԱՀ-ի գործառության արդյունավետությունը մեծապես կախված է այն բանից, թե ինչպիսին է ԿԾԱ-ի ճարտարապետությունը, խնդիրների և համակարգային ռեսուրսների ինչպիսի բազմություններ են ընտրված, ինչպես են կազմակերպված հաշվողական գործընթացները, կյանքի ցիկլի որ մոդելով է այն մշակված:

Կառավարման կամ տեղեկատվական ավտոմատացված համակարգերի ԾԱ-ի մշակումը կարող է իրականացվել կյանքի ցիկլի հետևյալ հիմնական մոդելներով [1].

- կատարել-ճշտել մոդելը,
- դասական կասկադային կամ ջրվեժային մոդելը,
- ձևափոխված կասկադային կամ հորձանուտի (ջրապտույտի) մոդելը,
- նախատիպավորման մոդելը,
- պարուրաձև մոդելը,

որոնք իրարից տարբերվում են մշակման ստանդարտ փուլերի ընդգրկման, միջանկյալ նախագծային լուծումների ստացման, սխալների հայտնաբերման ու վերացման, նախագծման ընդհանուր ժամանակահատվածի երկարության և այլ պարամետրերով: Առավել կիրառական և ծրագրավորման պրակտիկային ավելի մոտ է ձևափոխված կասկադային կամ «հորձանուտի» մոդելը (նկար 1): Կյանքի ցիկլի այս մոդելը թույլ է տալիս առանձին աշխատանքների զուգահեռ կատարումը, ինչպես նաև սխալի հայտնաբերման դեպքում՝ վերադարձ նախորդ փուլերին: Դա նշանակում է, որ այս մոդելում տեղի են ունենում ստուգումներ և գնահատումներ: Այս մոդելի յուրաքանչյուր փուլ համարվում է ավարտված միայն այն դեպքում, երբ ձևավորված են այդ փուլի համար ստանդարտով նախատեսված բոլոր փաստաթղթերը:



ԿՑ-ի ձևափոխված կասկադային մոդելը ամենահարմարն է համեմատաբար մեծ ԾԱ-ների մշակման համար, որոնք ունեն ճշգրիտ որոշված ֆունկցիաներ և որակի պահանջներ:

Համակարգի նախագծման փուլը սկսվում է առարկայական խնդիրների ձևայնացման, մեծ խնդիրները առանձին ֆունկցիոնալ մոդուլների բաժանելու (դեկոմպոզիցիայի) և առանձին մոդուլների մասնագրերն ու ալգորիթմները մշակելու աշխատանքներով: Ստեղծվում է առանձին մոդուլների բազմությունը, որոնց օգնությամբ ստեղծվում են համակարգի ֆունկցիոնալ խնդիրների ծրագրերը:

Ծրագրային մոդուլը ծրագրային միջոցի (ԾՄ) նկարագրության ցանկացած ֆրագմենտ է, որը ձևավորված է որպես ԾՄ-ի նկարագրման մեջ օգտագործվող առանձին ծրագրային արտադրանք: Դա նշանակում է, որ յուրաքանչյուր ծրագրային մոդուլ ծրագրավորվում, կոմպիլյացիայի է ենթարկվում և կարգաբերվում է ծրագրի այլ մոդուլներից անկախ և դրանով իսկ ֆիզիկապես բաժանված է նրանցից: Դեռ ավելին, մշակված յուրաքանչյուր ծրագրային մոդուլ կարող է ընդգրկվել տարբեր ծրագրերի կազմի մեջ, եթե կատարվում են այդ մոդուլի օգտագործման պայմանները, որոնք հայտարարված են նրա փաստաթղթերում:

Այսպիսով, ծրագրային մոդուլը կարելի է դիտարկել և՛ որպես ծրագրի բարդության դեմ պայքարի միջոց, և՛ որպես ծրագրավորման մեջ կրկնորդման

դեմ պայքարի միջոց (այսինքն՝ որպես ծրագրային գիտելիքների կուտակման և բազմակի օգտագործման միջոց):

Ոչ ցանկացած ծրագրային մոդուլ կարող է նպաստել ծրագրի պարզեցմանը: Լավ մոդուլը պետք է արտաքինից ավելի պարզ լինի, քան ներսից, իսկ այն օգտագործելը պետք է ավելի հեշտ լինի, քան կառուցելը:

Սակայն այսքան ընդհանուր հայտանիշների գործնական օգտագործումը բավականին պրոբլեմատիկ է: Դրա համար էլ ծրագրային մոդուլի կիրառելիության գնահատման համար Գ. Մայերսը [2] առաջարկում է օգտագործել մոդուլի հետևյալ կոնստրուկտիվ բնութագրերը՝ չափսը, ամրությունը, զուգակցումը (կցորդումը) այլ մոդուլների հետ, քարացածությունը, անկախությունը նրան դիմումների նախապատմությունից:

Հակիրճ նկարագրենք այդ բնութագրերը.

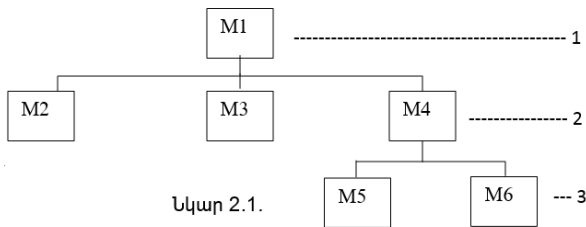
1. Մոդուլի չափսը նրա մեջ պարունակվող օպերատորների կամ տողերի քանակն է: Մոդուլը չպետք է լինի շատ փոքր կամ շատ մեծ: Փոքր մոդուլները բերում են ծրագրի մոդուլային կառուցվածքի ուռճացման և կարող են չվերադարձնել նրանց ձևավորման վրա կատարված վերադիր ծախսերը: Մեծ մոդուլները անհարմար են սովորելու և փոփոխելու համար, նրանք կարող են էապես երկարացնել ծրագրի կարգաբերման ժամանակը:
2. Մոդուլի ամրությունը նրա ներքին կապերի չափն է: Ինչքան մեծ է մոդուլի ամրությունը, այնքան շատ ներքին կապեր ունի այն և հետևաբար այնքան մեծ ներդրում ունի մեծ ծրագրի պարզեցման հարցում: Ինֆորմացիոն ամուր է համարվում այն մոդուլը, որը կատարում է (իրականացնում է) մի քանի գործողություններ (ֆունկցիաներ) տվյալների միևնույն կառուցվածքի վրա (ինֆորմացիոն օբյեկտի վրա): Այդ գործողություններից յուրաքանչյուրը մոդուլի մեջ ունի իր մուտքը և դիմելու իր ձևը: Ինֆորմացիոն ամուր մոդուլը կարող է իրականացնել, օրինակ, տվյալների արստրակտ տիպ:
3. Մոդուլի զուգակցումը (կցորդումը) այլ մոդուլներից ըստ տվյալների կախվածության չափն է, որը բնութագրվում է տվյալների փոխանցման եղանակով: Ինչքան թույլ է տվյալ մոդուլի զուգակցումը այլ մոդուլների հետ, այնքան ուժեղ է նրա անկախությունը այլ մոդուլներից:
4. Մոդուլի քարացածությունը նրա անկախությունն է նրան դիմելու նախապատմությունից: Մոդուլը կոչվում է քարացած, եթե նրան

դիմելու արդյունքը կախված է միայն նրա պարամետրերի արժեքներից (և կախված չէ նրան դիմելու նախապատմությունից): Մոդուլը համարվում է նախապատմությունից կախված, եթե նրան դիմելու արդյունքը կախված է այդ մոդուլի ներքին վիճակից, որը փոփոխվում է նրան նախկին դիմումների արդյունքում:

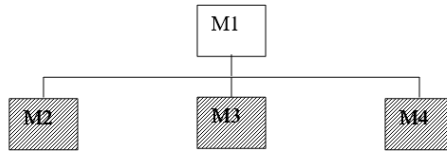
Ընդունված է նաև ծրագրային մոդուլին վերագրել հետևյալ հատկությունները.

1. Մոդուլը պետք է ունենա մեկ մուտք և մեկ ելք:
2. Մոդուլը պետք է լուծի առանձին խնդիր ըստ սկզբունքի՝ մեկ մոդուլ, մեկ ֆունկցիա:
3. Մոդուլի աշխատանքը չպետք է կախված լինի.
 - մուտքային տվյալներից,
 - այն բանից, թե որ ծրագրի համար են նախատեսված նրա ելքային տվյալները,
 - մոդուլը կանչելու նախապատմությունից:
2. Մոդուլը պետք է ղեկավարումը վերադարձնի այն ծրագրին, որը նրան «կանչել» է:
3. Մոդուլը կարող է «կանչել» այլ մոդուլների:
4. Մոդուլի չափսերը ցանկալի է սահմանափակել տեքստի 1-2 էջով:
5. Մոդուլը պետք է ունենա մասնագիր:

Գոյություն ունի մոդուլային ծրագրերի ստեղծման երկու հիմնական մեթոդիկա՝ վերից վար և վարից վեր կոդավորում ու թեստավորում: Ծանոթանանք վերից վար կոդավորման և թեստավորման մեթոդիկայի հետ: Դիտարկենք որևէ ծրագրային համալիրի կառուցվածքային դիագրամը (նկար 2.1): Կառուցվածքային դիագրամը երեք մակարդակի է և կազմված է 6 մոդուլներից: M1-ը ղեկավարող մոդուլն է: Համալիրի նախագծումը և կոդավորումը սկսվում է M1-ից: M1-ի թեստավորման և կարգաբերման համար պետք է ունենալ երկրորդ մակարդակի մոդուլները, բայց քանի որ դրանք դեռևս պատրաստ չեն, դրանց փոխարեն օգտագործվում են իրենց նմանակները՝ խցանները:

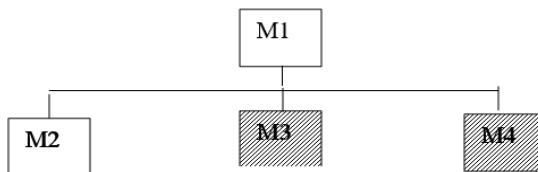


Քանի որ խցանների դերն այն է, որ ապահովեն վերին մակարդակի մոդուլների կատարումը, ուստի դրանք կարող են լինել բավականաչափ պարզ: Ուրեմն M1-ը կարգաբերելու համար M2, M3 և M4 մոդուլները փոխարինվում են իրենց խցաններով և կառուցվածքային դիագրամը կլինի (նկար 2.2)



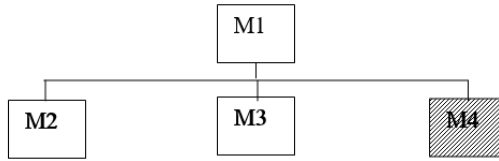
Նկար 2.2

Երբ M1-ի կարգաբերումը վերջացած է, սկսվում է M1-ի և M2-ի միացյալ կարգաբերումը, որի համար M2-ի խցանը փոխարինվում է M2 մոդուլով, իսկ M3 և M4 մնում են դեռևս խցաններ (նկար 2.3)



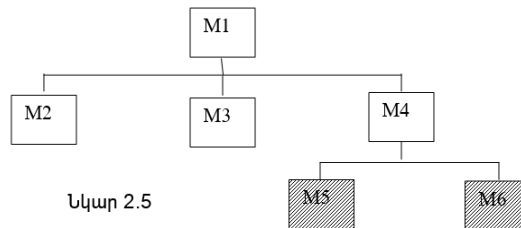
Նկար 2.3

Նույն ձևով են վարվում ծրագրային համալիրին M3 մոդուլը միացնելու ժամանակ (նկար 2.4)



Նկար 2.4

Որպեսզի միացնենք նաև M4 մոդուլը, պետք է ենթակա M5 և M6 մոդուլները փոխարինենք խցաներով (նկար 2.5):



Նկար 2.5

Նույն ալգորիթմով կարգաբերվում են նաև 3-րդ մակարդակի մոդուլները՝ M5-ը և M6-ը:

Ինչպես երևում է բերված դիագրամներից, յուրաքանչյուր մոդուլի կարգաբերման հետ միասին կատարվում է ամբողջ ծրագրային համալիրի կարգաբերումը վերից վար: Դրա հետևանքով ամեն մի նոր միացված մոդուլից հետո ունենք ծրագրային համալիրի աշխատող տարբերակ:

Նշենք, որ բազմամոդուլային ծրագրի յուրաքանչյուր մոդուլ, բացի իր հիմնական ֆունկցիաներից, իրականացնում է նաև մուտքային և կապի տվյալների ստուգումներ: Հետևաբար յուրաքանչյուր մոդուլ, այդ թվում նաև ղեկավարող մոդուլը, կարող է ունենալ կամ նորմալ կամ վթարային ելք: Նորմալ ելքի դեպքում կատարվում է վերադարձ կանչող ծրագրին կամ ըստ բազմամոդուլային ծրագրի սխեմայի՝ հաջորդ ծրագրին, իսկ վթարային ելքի դեպքում դիմում է գրանցման ծրագրին սխալը ֆիքսելու համար, որով և ավարտվում է բազմամոդուլային ծրագրի կատարումը: Գրանցման ծրագրին

դիմում է նաև բազմամոդուլային ծրագրի սխեմայի վերջին մոդուլը (նորմալ ելքի դեպքում)՝ մշակվող հաղորդագրությունը կամ հարցումը ժամանակագրական կարգով գրանցելու և արխիվացնելու համար:

Վերջում նշենք, որ ԾԱ-ի կյանքի ցիկլի ամենաերկարատև փուլերը իրականացման և փորձարկումների փուլերն են, որոնց ընթացքում ստեղծվում է ԿԱՀ-ի փորձնական նմուշը և իրականացվում են նրա կարգաբերման, փորձարկումների և ներդրման աշխատանքները: Հետևաբար վերից վար կոդավորման և թեստավորման վերը նկարագրված մեթոդի կիրառումը կարող է էապես կրճատել նշված փուլերի տևողությունները:

Քանալի քառեր՝ կառավարման ավտոմատացված համակարգ, ծրագրային ապահովում, կիրառական ծրագրային ապահովում, օպերացիոն համակարգ, ԾԱ-ի կյանքի ցիկլ, մոդուլային ծրագրավորում, վերից վար և վարից վեր կոդավորում և թեստավորում:

Օգտագործված գրականություն

1. Орлов С.А. Технологии разработки ПО. – СПб.: Питер, 2004.- 528с.
2. Майерс Г. Искусство тестирования программ. М.: Финансы и статистика, 1982. 176с.

SOME ISSUES ON SYSTEMATIC SETUP OF APPLIED SOFTWARE PROVISION OF MANAGEMENT AUTOMATION SYSTEM

H. A. MELIKYAN

Candidate of Economics, Associate Professor

Head of the Chair of Informatics and Mathematical Sciences

The method of top-down coding and debugging multi-module programs of applied software of management automation system is considered in the article.

It is noted that the bottom-up connection to the debugged program of separately developed modules leads to consideration of the entire program from the top to the bottom, which significantly reduces the time to assemble large multi-module programs.

О НЕКОТОРЫХ ВОПРОСАХ СИСТЕМНОЙ ОТЛАДКИ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННОГО СИСТЕМНОГО УПРАВЛЕНИЯ

Г. А. МЕЛИКЯН

Кандидат технических наук, доцент

Заведующий кафедрой информатики и физико-математических дисциплин

В работе рассматривается метод сверху - вниз кодирования и отладки многомодульных программ прикладного программного обеспечения АСУ: Отмечается, что снизу-вверх подключение к отлаживаемой программе отдельно разработанных модулей, приводит к рассмотрению всей программы сверху-вниз, что существенно сокращает время сборки больших многомодульных программ.