ИСТОЧНИКИ ОШИБОК В ПРОГРАММНЫХ СРЕДСТВАХ

Интеллектуальные возможности человека, используемые при разработке программных систем. Понятия о простых и сложных системах, о малых и больших системах. Неправильный перевод информации из одного представления в другое - основная причина ошибок при разработке программных средств. Модель перевода и источники ошибок.

2.1. Интеллектуальные возможности человека.

Дейкстра [2.1] выделяет три интеллектуальные возможности человека, используемые при разработке ПС:

- способность к перебору,
- способность к абстракции,
- способность к математической индукции.

Способность человека к перебору связана с возможностью последовательного переключения внимания с одного предмета на другой с *узнаванием* искомого предмета. Эта способность весьма ограничена - в среднем человек может уверенно (не сбиваясь) перебирать в пределах 1000 предметов (элементов). Человек должен научиться действовать с учетом этой своей ограниченности. Средством преодоления этой ограниченности является его способность к абстракции, благодаря которой человек может объединять разные предметы или экземпляры в одно понятие, заменять множество элементов одним элементом (другого рода). Способность человека к математической индукции позволяет ему справляться с бесконечными последовательностями.

При разработке ПС человек имеет дело с системами. Под *системой* будем понимать совокупность взаимодействующих (находящихся в отношениях) друг с другом элементов. ПС можно рассматривать как пример системы. Логически связанный набор программ является другим примером системы. Любая отдельная программа также является системой. Понять систему - значит осмысленно перебрать все пути взаимодействия между ее элементами. В силу ограниченности человека к перебору будем различать простые и сложные системы [2.2]. Под *простой* будем понимать такую систему, в которой человек может уверенно перебрать все пути взаимодействия между ее элементами, а под *сложной* будем понимать такую систему, в которой он этого сделать не в состоянии. Между простыми и сложными системами нет четкой границы, поэтому можно говорить и о промежуточном классе систем: к таким системам относятся программы, о которых программистский фольклор утверждает, что "в каждой отлаженной программе имеется хотя бы одна ошибка".

При разработке ПС мы не всегда можем уверенно знать о всех связях между ее элементами из-за возможных ошибок. Поэтому полезно уметь оценивать сложность системы по числу ее элементов: числом потенциальных путей взаимодействия между ее элементами, т.е. n! , где n

- число ее элементов. Систему назовем *малой*, если n < 7 (6! = 720 < 1000), систему назовем *большой*, если n > 7 . При n=7 имеем промежуточный класс систем. Малая система всегда проста, а большая может быть как простой, так и сложной. Задача технологии программирования - научиться делать большие системы простыми.

Полученная оценка простых систем по числу элементов широко используется на практике. Так, для руководителя коллектива весьма желательно, чтобы в нем не было больше шести взаимодействующих между собой подчиненных. Весьма важно также следовать правилу: "*все, что может быть сказано, должно быть сказано в шести пунктах или меньше*". Этому правилу мы будем стараться следовать в настоящем пособии: всякие перечисления взаимосвязанных утверждений (набор рекомендаций, список требований и т.п.) будут соответствующим образом группироваться и обобщаться. Полезно ему следовать и при разработке ПС.

2.2. Неправильный перевод как причина ошибок в программных средствах.

При разработке и использовании ПС мы многократно имеем дело [2.3] с преобразованием (переводом) информации из одной формы в другую (см.рис.2.1). Заказчик формулирует свои потребности в ПС в виде некоторых требований. Исходя из этих требований разработчик создает внешнее описание ПС, используя при этом спецификацию (описание) заданной аппаратуры и, возможно, спецификацию базового программного обеспечения. На основании внешнего описания и спецификации языка программирования создаются тексты программ ПС на этом языке. По внешнему описанию ПС разрабатывается также и пользовательская документация. Текст каждой программы является исходной информацией при любом ее преобразовании, в частности, при исправлении в ней ошибки. Пользователь на основании документации выполняет ряд действий для применения ПС и осуществляет интерпретацию получаемых результатов. Везде здесь, а также в ряде других процессах разработки ПС, имеет место указанный перевод информации.

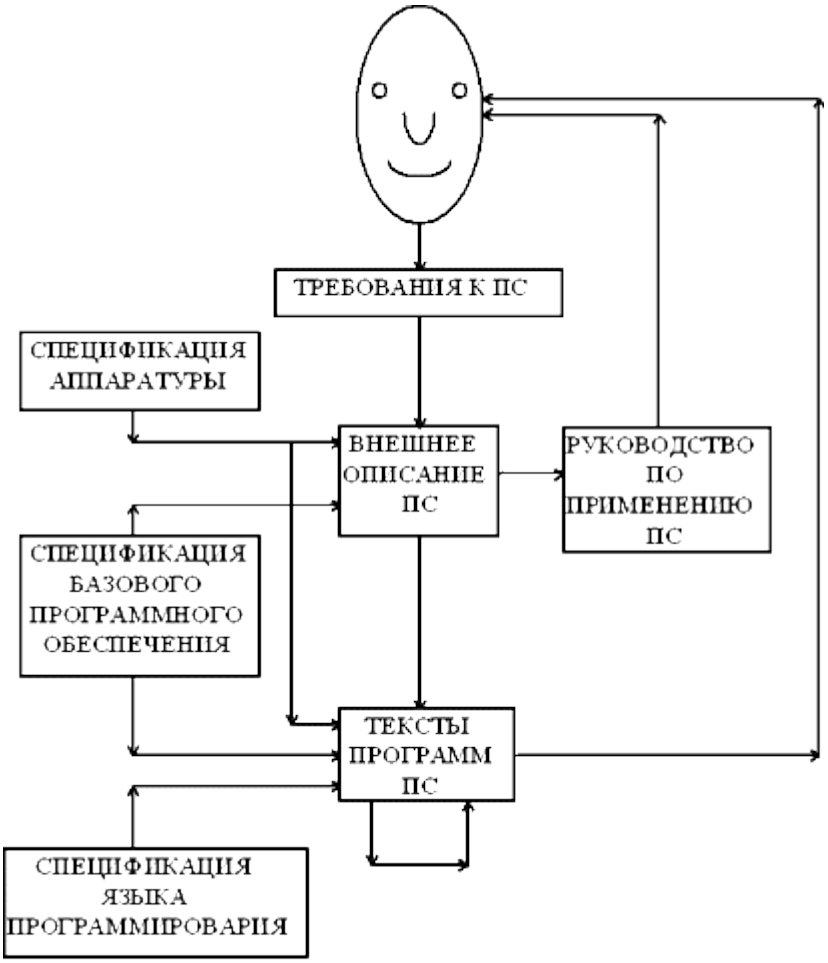

Рис. 2.1. Грубая схема разработки и применения ПС.

На каждом из этих этапов перевод информации может быть осуществлен неправильно, например, из-за неправильного понимания исходного представления информации. Возникнув на одном из этапов ошибка в представлении информации распространяется на последующие этапы разработки и, в конечном счете, окажется в самом ПС.

2.3. Модель перевода.

Чтобы понять природу ошибок при переводе рассмотрим модель [2.3], изображенную на рис.2.2. На ней человек осуществляет перевод информации из представления А в представление В. При этом он совершает четыре основных шага перевода:

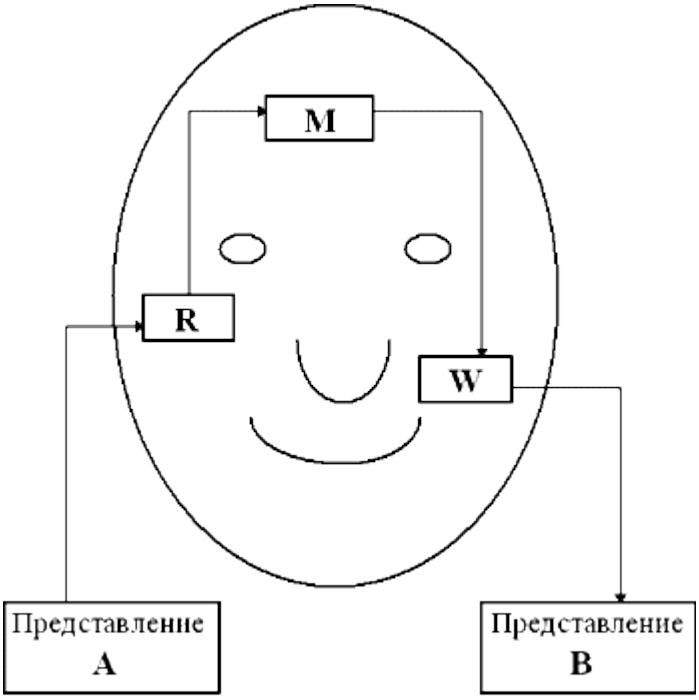

Рис.2.2. Модель перевода.

- он получает информацию, содержащуюся в представлении A, с помощью своего читающего механизма R;
- он запоминает полученную информацию в своей памяти M;
- он выбирает из своей памяти преобразуемую информацию и информацию, описывающую процесс преобразования, выполняет перевод и посылает результат своему пишущему механизму W;
- с помощью этого механизма он фиксирует представление B.

На каждом из этих шагов человек может совершить ошибку разной природы. На первом шаге способность человека "читать между строк" (способность, позволяющая ему понимать текст, содержащий неточности или даже ошибки) может стать причиной ошибки в ПС. Ошибка возникает в том случае, когда при чтении документа A человек, пытаясь восстановить недостающую информацию, видит то, что он ожидает, а не то, что имел в виду автор документа A. В этом случае лучше было бы обратиться к автору документа за разъяснениями. При запоминании информации человек осуществляет ее осмысливание (здесь важен его уровень подготовки и знание предметной области, к которой относится документ A). И, если он поверхностно или неправильно поймет, то информация будет запомнена в искаженном виде. На третьем этапе забывчивость человека может привести к тому, что он может выбрать из своей памяти не всю преобразуемую информацию или не все правила перевода, в результате чего перевод будет осуществлен неверно. Это обычно происходит при большом объеме плохо организованной информации. И, наконец, на последнем этапе стремление человека побыстрее зафиксировать информацию часто приводит к тому, что представление этой информации оказывается неточным, создавая ситуацию для последующей неоднозначной ее интерпретации.

2.4. Основные пути борьбы с ошибками.

Учитывая рассмотренные особенности действий человека при переводе можно указать следующие пути борьбы с ошибками:

- сужение пространства перебора (упрощение создаваемых систем),
- обеспечение требуемого уровня подготовки разработчика (это функции менеджеров коллектива разработчиков),
- обеспечение однозначности интерпретации представления информации,
- контроль правильности перевода (включая и контроль однозначности интерпретации).

Литература к лекции 2.

2.1. Э. Дейкстра. Заметки по структурному программированию// У. Дал, Э. Дейкстра, К. Хоор. Структурное программирование. - М.: Мир, 1975. - С. 7-19.

2.2. Е.А. Жоголев. Технологические основы модульного программирования. //Программирование, 1980, #2. - С. 44-49.

2.3. Г. Майерс. Надежность программного обеспечения. - М.: Мир, 1980. - С. 22-28.