# Lawson Application Customization

## Managing Changes in Lawson Applications

Milo Tsukroff
Edited by Paul Warner

November 2006

# Lawson Application Customization
## Managing Changes in Lawson S3 Applications

### *Introduction*

Lawson's S3 applications deliver most of what companies need for back office functions. However, they do not always fulfill all business requirements. Customizing Lawson applications for non-standard functionality is possible, since Lawson delivers source code to customers. There are several methods of doing this. Each method has benefits, but each also has immediate and long-term costs.

There is a right way and wrong way to customize Lawson. Like the 'butterfly effect', small changes can ripple out, causing major disruption years later. This document details each method of customizing Lawson.

Customizations quickly become expensive to maintain. To make tracking changes manageable, a simple, workable documentation system was developed. This is useful both for tracking changes and for estimating how much work those changes will eventually cost. This documentation strategy will also give you the total cost of all of your Lawson customizations. It provides metrics for planning, as well as tracking capabilities to measure progress in reapplying customizations.

This approach is not an academic exercise. It is based on practical experience, both good and bad, in private industry, including lessons learned from a site where Lawson was customized into oblivion. When it was time to do a major upgrade the extensive changes doubled the upgrade cost. We could not upgrade after that.

My department has used this approach for over two and a half years at Private Healthcare Systems. It has been used to estimate the time needed for several CTP's, an MSP, and a complete new Product Line install. It reduced the complexity of reapplying customizations, gave good predictions of labor costs, provided metrics for planning, and accurately tracked progress.

This document focuses on the “core” of the standard Lawson applications, specifically those affected by CTP's, MSP's, and Upgrades. It does not cover the web layers (IOS, Portal, ESS, MSS, VSS, Design Studio, ProcessFlow, etc.) many of which are never customized, or which are specifically designed for customization by Lawson customers. They are not discussed here. [In the specific case of ProcessFlow, source code customizations needed to activate ProcessFlow can be tracked using this approach, if desired.]

The author is familiar with UNIX and Windows environments, with underlying source code constructed in COBOL. Lawson software using RPG is not addressed. It is the hope of the author that Lawson clients in the IBM world can still get benefit from this document.

*SECTION 1*

# Customizing Lawson S3 Applications

### *Lawson's Policy on Customizations*

Lawson will not support any program or data structure that is changed or created by a client. Therefore, all changes that a customer makes must be supported internally or by an expensive consultant. This means that solutions for business requirements should always seek to leave Lawson functionality unchanged as much as possible. Solutions that require changes to Lawson data structures or programs have a direct impact to long-term internal support costs. They also increase time and expense when going through required Lawson maintenance and upgrades.

Lawson has been addressing the serious maintenance issues that many types of customization introduced in the past. The Application Maintenance Toolkit (AMT), introduced in 2005, provides new tools that greatly increase flexibility in maintaining customizations, and thus markedly reduce the associated costs. The documentation developed in 2005 was updated to take the AMT into account.

### *Lawson Change Levels*

Lawson makes application changes in two levels of scope: CTP's and MSP's (version point change), and Upgrades (version number change). These mandated changes are necessary for continued Lawson support.

Each change can involve a number of areas, affecting databases, source code, and other objects. Every area and what they contain are examined for customization impact.

### *The Lawson Metamodel*

The Lawson metamodel is the aggregation of all information on how the Lawson software and data structures are constructed and related. This information is kept in the GEN database tables. The metamodel is a higher logical layer than the DBMS. The actual data structures used in a DBMS's underlying SQL varies depending upon which DBMS is used.

Querying the GEN tables can be very informative; directly modifying them is dangerous and not supported by Lawson. The approach to customization described here leaves them strictly alone.

### *Data Modifications*

**Naming Conventions:** Relational database systems management systems (DBMS's) such as DB2, SQL Server, and Oracle use the terms table, row, and column. Lawson uses the terms file, record, and field, respectively. This table shows how their naming conventions are related.

| **Lawson** | **SQL** |
|---|---|
| Product Line | Database |
| File | Table |
| Record | Row |
| Field | Column |

The Lawson-sanctioned method of adding columns to a table is to use the **dbdef** tool. This ensures that all required Lawson program resources are automatically available in the Lawson 4GL environment, and maintains the integrity of the metadata model.

New tables for holding non-standard data can be created with **dbdef**. These are linked to standard Lawson tables by indexes, conditions, and relations.

Lawson tables can be directly modified. New columns can be added to a table with **dbdef**. Also, columns can be used for purposes other than their Lawson intended purpose. This is done to preserve Lawson standard table structures when new columns are needed for non-standard data.

Work files are created using an entirely different tool, **workdef**. These files are used for temporary purposes, such as sorting and data transfer. They are never part of the overall schema that the DBMS sees.

Data modifications should always be done using the Lawson tools. This ensures that changes fit right into the Lawson environment. Even so, Lawson tables can be modified directly in the DBMS. Any modifications made this way are invisible to Lawson. Accessing Lawson data files without using Lawson is only recommended for reporting, or interfacing changes is data to external systems..

When changes can be made using the DBMS without disturbing the Lawson metamodel, it is one way to avoid creating a custom program. It is still a dangerous thing to do, requiring careful analysis to avoid problems.

**Work file** changes are not severe in the near term. Usual changes are: Making adjustments to sorts; modifying in/out data transfers; and other temporary file uses. They do not have a high impact on long-term maintenance either. Changes to Lawson-supplied work files can be easily reapplied when an upgrade wipes them out. Custom work files will usually not be affected.

Work File **CSV Attribute** changes are small but vital. CSV work files can be customized by defining file attributes that change both the way that CSV files are interpreted, and more importantly, the name and path for incoming and outgoing files. These changes do not take much time, and they make life a lot easier in a Lawson

environment. However, they often do not get propagated properly, so it is important to track them. Long-term impact is low.

**Field Re-Use** is necessary when user data fields are all used up, or simply not available. Re-using a field requires source code changes. Lawson upgrades can affect this method also, so they always have to be checked. In the near term, Field Re-Use ranges from very easy to moderately difficult. Long-term, Field Re-Use requires both that program changes be checked again, possibly being reapplied, and that the field selected be checked again. Therefore, the degree of difficulty that a Field Re-Use requires on the near term corresponds to its Long-Term maintenance costs. More difficult changes require more work "down the road", because they add manual work to applying CTP's, MSP's, or Updates.

**Customized Lawson Tables** are when new fields are added to a standard Lawson table, or when changes are made to field definitions. This is a very "quick and dirty" method of getting changes into Lawson. Immediate costs are very low. Long-term, since Lawson will not support such changes, maintenance costs become higher.

For immediate needs, the extremely low cost of modifying tables makes it very attractive. This is a common method used by consultants. In the long term, the AMT functions **metadumptbl** and **metaloadtbl** makes it fairly easy to bring changes forward into tables being updated by a CTP, MSP, or Update. They do add manual work when applying these "automatic" Lawson changes, though.

**New Custom Tables**, **new indexes** for existing tables, and **new work files**, can be created when needed. Since these take time and research to develop, they are more time consuming up front. However, since Lawson upgrades never touch them, they take much less time to maintain. The long-term impact is low, since they are not affected by Lawson changes of any kind. The only thing to check, usually, is changes to referenced key fields.

**SQL Changes**, such as queries or triggers, do not affect Lawson. They are a very good way to handling outgoing data transfers, and doing reporting. Short-term, there is low cost and no impact upon Lawson source code. (However, there can be an impact on performance, and extensive triggered logic can create timeout errors.) There is a long-term impact, though it is not severe: When Lawson table changes occur, SQL queries must be reevaluated. Lawson adjusts how it uses fields from time to time, which can affect customer-developed queries and extracts.

Because the actual data structures used in a DBMS's underlying SQL varies depending upon which DBMS is used, SQL Changes are specific to the DBMS used.

**New elements** are often needed to create fields in new custom databases. These are easy and quick to add when creating new fields in dbdef. In the long term, the new AMT tools **metadumpelm** and **metaloadelm** allow maintaining custom elements quickly.

### *Program Customizations*

Adding new functionality to Lawson requires that programs be customized or added. Existing Lawson programs can also be customized directly, or new programs can be created.

The Lawson-recommended method of customizing an existing program is to make a clone of the program with a client-specific name, leaving the original Lawson program alone. Modifications are then made to the clone. Access to the original program is shut off, but it is retained and is kept current with CTP's, MSP's, updates, and upgrades.

Entirely new programs can be written using Lawson's 4GL tools. This is a 'clean' method of adding functionality, as Lawson changes do not touch customer-written programs at all.

When business logic or data tracking requirements go beyond Lawson-supplied functionality, user fields in many important files can be used. Although Lawson has supplied user fields in many files, and using these fields does not affect Lawson data structures, programs and reports are often not designed with specific access to them. Therefore, it is often necessary to make Lawson program changes in order to use them. Also, there is precedent for Lawson using previously untouched user fields.

Cloned programs have various maintenance issues, depending upon the nature of the change. The location of the change also determines what maintenance issues will occur.

Lawson's proprietary 4GL program development environment maintains a number of objects for every program: Screen Definition Form (.scr); Report Definition Form (.rpt); Program Definition (PD); Working Storage (WS), Workfiles (.wrk), Messages (.msg), and up to three User Exit programs. The compiler assembles these specialized components into Lawson-specific COBOL (RPG is not covered in this document). Program customization has differing impact depending upon which component is modified.

Other objects that impact programs, which supply programming items to multiple programs, are: Screen Rules (.sr), Object Rules (.or), Key Numbers (KN), and System Key Numbers (SKN).

**Screen Definition Forms (.scr)** are usually relatively easy changes. These can range from the most complex, down to simple relabeling for user fields. The complexity of screen changes is determined by the scope of the changes being made.

Long-term, these changes are easy to re-apply. Even extensive changes can be applied quickly.

This is the only customization which can easily be done to a regular Lawson program. When a simple label change is needed, it is much too much trouble to create a clone program. Instead, this change can be quickly reapplied to the original Lawson program after a Lawson CTP or update changes it.

When making major changes to a screen, it is best to copy the program into a custom clone program, and make the changes there.

Documenting is especially important when making small changes. Otherwise, it is easy to lose track of the minor labeling changes that users find so useful in their jobs.

**Report Definition Forms (.rpt)** have about the same impact as .scr's. Again, the scope of changes determines the complexity of the changes. The long-term hit is low. This type of change is often not done at companies that use direct SQL queries or external reporting tools.

**Procedure Definition (PD)** changes are the most powerful, and the most costly, changes that can be made. These are necessary for field re-use, changes to business logic, new tables and columns, and a myriad of other reasons. They can range from relatively small to absolutely major.

No matter how small the change, the long-term costs of PD changes is even higher than the initial cost. Maintenance on these internally-developed custom programs is a direct expense, since they are not covered by the Lawson support contract. When Lawson upgrades occur, major research is necessary to determine what impacts there are. (Consultants love to make this kind of change, by the way.)

Sometimes Procedure Definition changes must be made in subprograms, that is, procedures stored in the pdlib folder. This type of change can affect many programs. When making a change to a subprogram, it is essential to track and re-compile all programs affected. Sometimes the impact of a subprogram change is too large to be safely attempted, and another way has to be found.

**Working Storage (WS)** changes are often necessary when making Produdure Definition changes. Their impact and costs are about the same as screen and report form changes. There is an up-front cost for these changes, but usually changes are just minor additions. While changes are sometimes extensive, they can all be re-applied quickly after a Lawson change wipes out the changes.

**New Programs** can be written by using the Lawson 4GL tools. The tools integrate with Lawson's database tools and Lawson-specific library routines to allow a programmer to develop programs much more quickly than with traditional COBOL.

The initial development phase is a relatively high cost. Long-term, they are easy and inexpensive to maintain. Lawson upgrades minimally impact them, since they are not linked to any Lawson programs. Long-term maintenance costs are somewhat increased, though, by the research required to verify that links still work. Minor tweaks may still be necessary.

**User Exit programs** are a Lawson-sanctioned way to add customization to standard supplied programs. These can add a variety of business rules and logic. The cost of developing these is very similar to writing new programs. The advantage of user exits is that both up-front development costs and long-term maintenance costs are lower than with an entirely new program.

User Exits are the recommended method to provide hooks into ProcessFlow flows.

**Key Numbers** and **System Key Numbers** tie screens to database fields and provide a 'hook' for Selects and Drill-Arounds. Creating new Key Numbers is sometimes necessary, especially when creating Selects and Drill-Arounds for new tables. Long-term, it is usually only necessary to re-enter them when a new Lawson environment is set up. The new AMT tools **metadumpskn** and **metaloadskn** are very helpful in maintaining custom key numbers.

**Screen Rules** and **Object Rules**, which reside in the .sr and .or files respectively, define Selects and Drill-Arounds. Making modifications to these files is quick and easy. Since there is only one of each of these files in each System Code, in the long term, changes can become cumbersome to maintain. Careful and complete documentation can help to keep maintenance time to a minimum.

**Libraries** are collections of routines that are called by Lawson programs. Because of the scope of impact of a library routine change they should not be attempted. They are relatively quick and easy to do up front, therefore an easy change for a consultant. In the long term, they can be very time-consuming to maintain due to the necessity to manually check them.

***Other Customization Areas***

There are some other customization areas that involve the Lawson Environment, but are not related to files or programs. These are, New System Code, Menu customization, Help Text changes, Security Settings, Jobdef settings, and Language File changes.

Creating a **New System Code** is a clean way to create an area for custom databases and programs. It does not add a significant time to development efforts. In the long term, it makes maintenance easier. Since Lawson changes will never be applied to your own custom system code, long-term maintenance is quite low.

It is important to create a system code which will not conflict with any system code that Lawson creates. One method that I have used successfully is to always start the system code with the letter "Z".

**Menu customization** is quite common. This directly affects users who don't use F8 to move between screens. Therefore changes should be maintained diligently. The initial cost of putting in menu customizations is fairly low. The new AMT 8-5 tools metadumpmnu and metaloadmnu will help to keep the long-term maintenance cost low as well.

**Help Text changes** can be done and maintained by multiple tools. Easy to dump, edit, and reload.

**Security** – Setting up Lawson security is a necessary and time-consuming task, in the beginning. Once it is fully set up, few changes are needed. It is sometimes necessary

to make security changes as part of a customization, and therefore they should be noted. Changes via CTP's, MSP's, and most Upgrades will not affect security. (The changeover from **laua** security to Lawson 9 LDAP-based security is an exception.)

The security systems within Lawson, whether laua-based or Lawson 9 LDAP-based, is powerful, comprehensive, and detailed. That makes security maintenance in Lawson always time-consuming. Tracking changes needed for customizations is helpful.

**Jobdef** – Setting up Lawson often requires that jobs be set up for special purposes. Once jobs are entered in, they are usually set for the life of the Lawson system. However, customizations that affect programs called from jobdef entries can impact how those jobs run. It is important to track the impact to jobs when putting a customization into place.

### *Outside Lawson*

Many times, writing new programs can be avoided entirely by using software that queries Lawson tables. This is easiest when creating reports. A number of companies have used **SQL extracts** very successfully. **External reporting tools** like ReportSmith or Crystal Reports, which get their data directly from Lawson tables via SQL, are also a good solution.

Even though SQL extracts and external reporting tools are not directly affected by Lawson changes, they can be affected by changes in how Lawson tracks data. From time to time, Lawson changes how data is linked or where data is stored. These changes can affect reporting. Therefore, any time a change is put into Lawson, the external extracts and reports should be checked.

### *Data Interfaces*

In my experience, there is no accounting system that exists entirely without data interfaces to the outside world. This is such an important consideration that I need to address it briefly in its own section here.

**Lawson-Standard Interfaces** work very well. A great deal of customized code can be developed in external systems, which are simply interfaced to Lawson. This is a clean way to replace Lawson functionality without changing Lawson itself. In one company, payroll time records are extracted via rngdbdump, massaged, reported on, consolidated, and then re-loaded for payroll via PR530 "Time Record Interface CSV".

**Customized Interfaces** are quite complex in their impact. Outgoing interfaces require writing either a custom Lawson program, or special SQL extracts. Incoming interfaces are much worse. When a business need requires creating a special interface into Lawson, it involves a considerable cost for initial development. Long-term maintenance costs can also be quite high. It is possible to greatly reduce long-term costs at development time by making a custom interface as rugged as possible. A custom interface has to be re-evaluated every time a Lawson change comes through.

Adding too many customized interfaces helped doom my prior company's prospects for a successful Lawson upgrade. The time needed for maintaining them

should be carefully considered when planning to use a non-Lawson solution. Several vendors specialize in providing custom Lawson interfaces. They may be a more cost-effective solution in the long run than creating your own customized interface.

### *Impact Scope of Lawson Changes*

Lawson changes occur in three levels of scope: CTP's and MSP's; Updates; and Upgrades. These mandated changes are necessary for continued Lawson support. Creating new Product Lines can also require re-applying all customizations, if a "clean" Product Line install is desired.

**CTP's** are issued to fix bugs, address defects in Lawson functionality, or make changes required by users. These are usually limited in scope. **MSP's** are collections of CTP's. They bring a Lawson installation up to date on all of the CTP's issued since the last Update or MSP.

An **Update** will bump up your Lawson level one point. For example, 8.0.2 to 8.0.3. This is a more significant change. Usually, all or mostly all modules are touched to some degree. All changes in every licensed module have to be evaluated. The time required for analysis can be extensive. All customized programs and tables affected have to be updated or re-compiled as part of the Point Upgrade.

An **Upgrade** is basically a version change, when Lawson goes through a level number change, such as going from Level 7 to Level 8. This is a "full court press", because every module that a customer has licensed will be involved.

A **new Product Line install** involves a "clean" Product Line install, putting in the plain vanilla Lawson applications, then re-apply all of your customizations. This can be time-consuming. The benefit of doing this is that you end up with a Product Line that has an unquestioned pedigree. There are no questions about anything that has been changed.

Of course, you can also take the shortcut of simply copying over the COBOL source code from existing source directories to the new Product Line's source directories. This does not bring over other objects, such as jobdef entries or menus, so if you take this shortcut, you must remember to bring everything over else that doesn't directly copy over.

The actual amount of updates required to maintain customizations depends upon the actual changes. Sometimes, CTP's and MSP's can have a significant impact and require a great deal of work, when they hit an area where Lawson has been heavily customized.

### *Troubleshooting Problems*

Problems may occur with a customization after a CTP, MSP, or Update. There is precedent to Lawson making changes to undocumented sections of source code which unexpectedly affect customizations you didn't think would be involved. Check the preview.log to find affected programs. Don't be surprised to find that more customizations than you expected are affected by what appeared at first glance to be a simple fix.

Good **Customization documentation** such as presented here as the ***Lawson Change Control*** system is also a good defense against potential problems. There's nothing better than a full explanation of how the customization works.

**Use the *diff* command** to compare source code between before-and-after snapshots of the source code directories. The diff output will show you exactly what Lawson changed. In case of unexpected problems it may point up what is causing the trouble.

In the case of cloned programs, you should clone the program again from the source program in order to compare source code. This is tricky, as text inside a program is also changed in the cloning process. You need to copy the updated program to a clone of the same exact name. Make certain that you can restore the original customized clone, or that you have a good copy in another Product Line.

**User Test Plans** are needed whenever affected programs are updated. They must be ready for when the next CTP or MSP impacts a customization. Usually users or a Quality Assurance team run test plans. Checking on the impact of Lawson-mandated changes insures that minimal problems will occur in production Product Lines.

**Getting Help from Lawson** takes a bit of careful maneuvering. Since Lawson will not support customizations, when problems with a customized program occur, the GSC is not supposed to give you any help. To get help from the GSC, you must go back to the source program, make certain that it is compiled properly, and reproduce the problem with the "clean" program. Then you can either look up the problem on the Knowledge Base, or place a support call. At no time must you refer to the customized program, since Lawson cannot help you with that.

Once you have an answer, the fix may involve changes in one or more of the many programming objects. Be certain to take a copy of the original customized program's source code before running the supplied CTP. After the fix has been applied, compare source code objects to see what changes were done. The changes must then be applied to all clone programs. When straight Lawson programs have been slightly modified (such as relabeling screen fields), the customizations can be reapplied to the Lawson program.

If all else fails, Lawson will gladly supply a consultant who will work with a customer to figure out what is going wrong. This is billable time and is not covered in the support contract.

*SECTION 2*

# A Working System for Managing Lawson Customizations

## *Managing Change*

Custom changes in Lawson must be tracked both completely and effectively. Detailed notes about the change itself must be made at the time of the change. They must be complete enough so that the change can be reproduced again, without time wasted for research. They must be effective enough to allow a reviewer to determine the amount of work needed. The change documentation should primarily focus upon the actual changes to Lawson tables and programs. Technical notes and detailed business reasons, which should be kept track of, can be included as appendices.

The documentation architecture presented here is a two-tiered tracking mechanism for customizations to Lawson. There is a tracking document, which provides details of each change. Then there is an overview spreadsheet, for summary information. This provides recording exactly what was done for each change, and also gives a quick overview of changes-to-date, for tracking and planning purposes.

**Lawson Change Control** documents, written in Microsoft Word, have specific information on each change. These are numbered sequentially - LCC-001.doc, LCC-002.doc, etc. These must contain enough information to be able to reconstruct the customization. Any special efforts needed to put a change into production must be noted. The sections 'Pre- and Post Testing Process' and 'Back Off/Recovery Plan' are especially important and provide completeness to the change process.

The document **LCC-nnn-Master.doc** is a blank LCC document. Copy it, rename it, and fill it in for each change. **LCC-001-example.doc** is an example LCC document.

A separate folder should be made for each LCC number. This gives the developer a place for the LCC document and any working files necessary.

The **Overview Spreadsheet**, "Lawson Change Control Master Listing" "**LCC-Master-List-empty.xls**", a Microsoft Excel spreadsheet, gives a complete view of the scope and complexity of accumulated changes. Each line in the overview spreadsheet references one LCC number. Fill in the line for each LCC based on what is filled in on the LCC document. The checkoffs in the Object Changed box in the LCC document must be duplicated by "X"'s in the appropriate boxes in the spreadsheet.

Once the information for an LCC is filled in correctly, the Hours Calculation column computes how much work each customization will cost for maintenance. The calculation adds a certain amount of time for each customization checked with an "**X**". The formula uses the weighting for each type of customization, multiplies program changes by the factor for New Programs, and multiplies the entire sum by the severity factor. The gives you a quick evaluation of the total impact of Lawson customizations. The **total at the bottom right** gives you the total for all LCC's that you have applied to

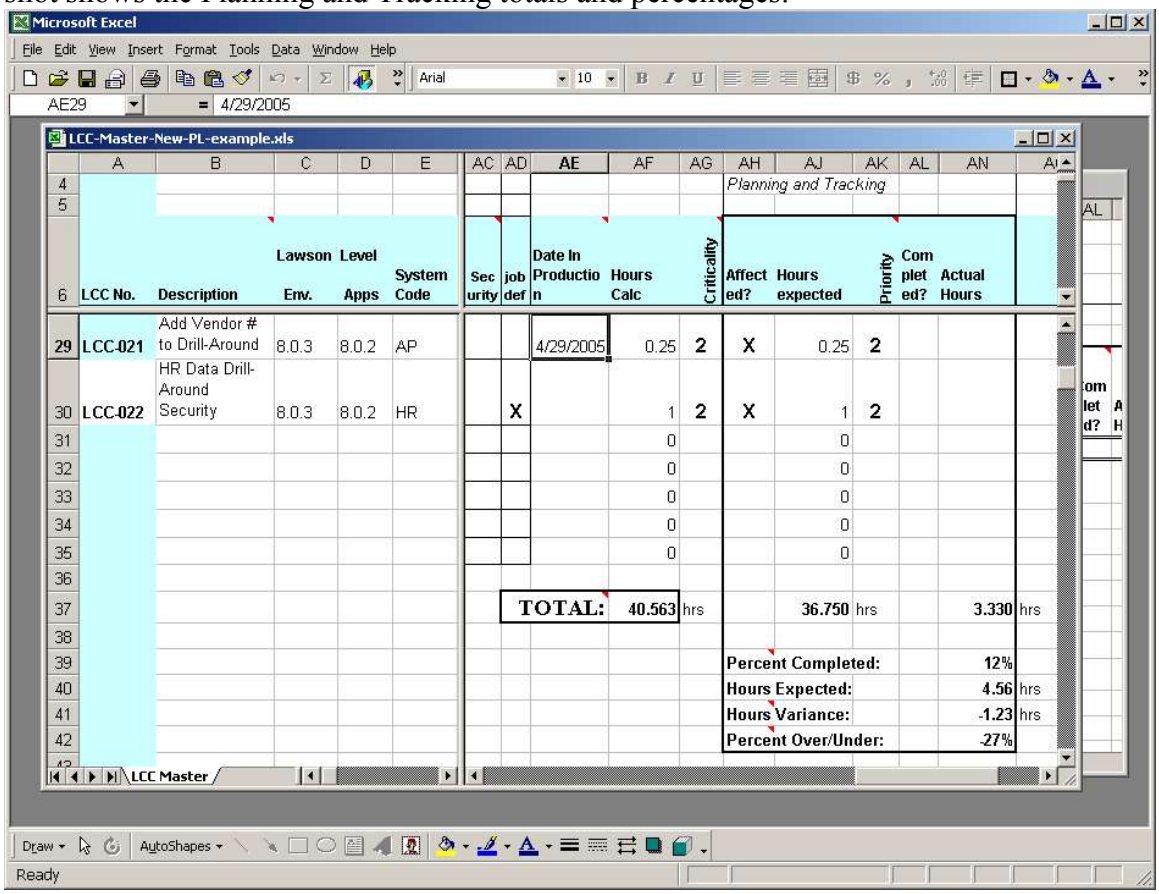
your system. If you have captured all of your changes in LCC's, this is the *total hours all of your Lawson customizations will cost you.*

The values that I have supplied have worked very well for us. We have found that they are within 30% of the actual time needed. You may find that you have to adjust the calculation for your own working group. The "**Weighting**" row gives you a quick and easy way making first-approximation adjustments without changing the calculation formulas. The "**Severity Multipliers**" section at the very top allows you to make adjustments to the amount that each severity level multiplies the overall time. Further refinements will require a skilled programmer to adjust the formulas.

### *Planning For a Change*

When a Lawson change such as a CTP, MSP, or major upgrade is applied, the section on the far right labeled "**Planning and Tracking**" comes into play. First, make a copy of your current Master spreadsheet. The copy becomes your tracking spreadsheet for the duration of applying the change.

The spreadsheet "**LCC-Master-New-PL-example.xls**" is an *example working spreadsheet*. This was used when setting up a new Product Line. The data is "almost real". Only a few changes were made to maintain confidentiality. PLEASE DO NOT MODIFY AND USE THIS SPREADSHEET, because it is not up to date. This screen shot shows the Planning and Tracking totals and percentages:



To fill out a Planning and Tracking section, first a programmer determines which LCC is affected by the Lawson change. For each LCC that is affected, put an "**X**" in the "**Affected**" column. When completed, the **total expected hours for the change** is given at the bottom.

The next step is to set the priorities for the work. These can be the same as the Criticality column. If needed, depending upon the exact nature of the Lawson change, or the time of the month, the priorities may have to be adjusted.

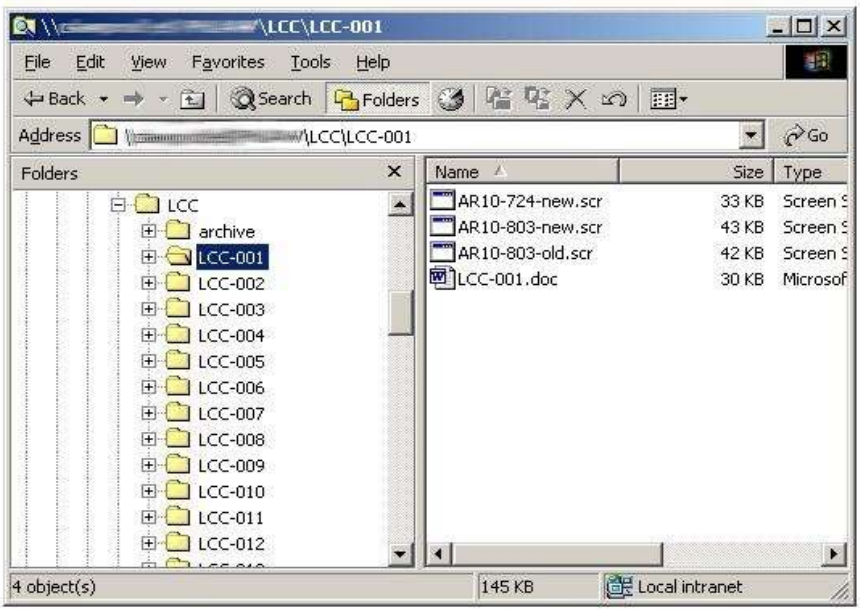
With the total expected hours, and priorities set, a manager now has an action plan for applying Lawson customizations.

As each LCC is completed, check it off with an "**X**" in the "**Completed**" column, and fill in the actual hours. Metrics that automatically fill in are, **Percent Completed**, **Hours Expected** (what time the jobs should take), **Actual Hours**, and **Percent Over/Under**. These will help a manager to see how far along the project is, how much total effort has been made, and how accurate the estimate of time is.

Since the estimate of time is based on a formula that I set up by trial and error, you may have to make adjustments. Adjust each LCC by adjusting its letter in the "**Severity**" column. If there is more or less time needed for changes to individual objects, such as menu changes always taking longer than expected, adjust the Weighting for that object. If there is an overall bias, then you may have to adjust the Severity multipliers. Please do not assume that my formulas will necessarily work for your company. Feel free to adjust things.

### *Storing LCC records*

Keep the overview spreadsheet in the folder that contains all of the individual LCC folders. This allows quick access to the summary, and to individual LCCs as needed. Here is a screen shot of our LCC folder:



## Conclusion

With each new business requirement, Lawson may require customization. Any and all of the methods laid out here can be used. With the understanding of each method's impact on both immediate and long-term costs, management should be able to determine the best method to meet every requirement.

As changes build up, keeping Lawson Change Control information up to date to will continue to keep customization costs under control. When required Lawson changes are applied, or when an upgrade is being planned, management can see what manpower is needed to reapply customizations. This makes it relatively easy to budget for the needed changes. Tracking progress will help to keep your Lawson system under control.

AUTHOR:

Milo Tsukroff
milotsukroff@vh.net

COPYRIGHT:



FREEWARE LICENSE:



DISCLAIMER: