8

الفصل الثامن
هندسة برمجيات المقررات ونظم المعلومات التعليمية

1 - المقدمة

2 - التصميم التعليمى

3 - هندسة البرمجيات

4 - هندسة برمجيات المقررات التعليمية

5 - نماذج هندسة برمجيات المقررات التعليمية

6 - إعداد النمذجة التمهيدية

7 - الوسائل/ الوسائط التمهيدية

8 - أدوات تطوير البرمجيات

9 - الخلاصة

1 - المقدمة

يقدم هذا الفصل نظرة شاملة لدورة حياة هندسة برمجيات المقررات ونظم المعلومات التعليمية Courseware and Information Systems Engineering كمقدمة تستخدم بالتفصيل لدراسة أجزاء نظام المعلومات التعليمى التى ترتبط بمكونات تطوير برمجيات المقررات التعليمية التى يمكن الاستفادة منها فى مساندة صناعة تكنولوجيا المعلومات من حيث تطوير البرمجيات التعليمية التى تتفق وتتواءم مع السوق التعليمية المصرية بصفة عامة والعربية بصفة خاصة.

وفى الوقت الحالى، يلاحظ عدم توافر طريقة أو وصفة واحدة يمكن استخدامها فى تطوير برمجيات المقررات التعليمية حتى تصبح مقبولة على نطاق دولى وتستخدم فى كل الظروف. وبدلا من ذلك، يمكن تحديد نمط عريض يستوعب معظم نماذج أو مخططات تطوير البرمجيات الممكن الاعتماد عليه فى الظروف المعينة التى تطبق فيها.

وفى محاولة جعل التنوع فى نماذج ومخططات هندسة تطوير البرمجيات مفهوما، فإن هذا العمل يوصف هندسة البرمجيات كمزيج موحد لمجالين أساسيين متفق عليهما من قبل المتخصصين. المجال الأول يختص بهندسة البرمجيات كعملية تطوير إعداد البرمجيات، أما المجال الثانى يتعلق بالتصميم التعليمى كعملية تطوير التعليم للإمداد والإتاحة بواسطة الحاسبات الآلية أو بأساليب أخرى. وعند استعراض التصميم التعليمى، استخدمت نظرية واحدة ببعض التفاصيل التى بينها واستعرضها روبرت جاجنية (Robert Gagne, 1995) لتوضيح أهمية نظرية التعلم المحددة لنموذج التصميم التعليمى.

ويوصف هذا العمل باختصار هندسة البرمجيات ويدمجها مع التصميم التعليمى للتوصل لهندسة برمجيات المقررات التعليمية. بالإضافة لذلك، يناقش ويوصف خطتين منشورتين لهندسة برمجيات المقررات التعليمية، إلى جانب مخططات أو نماذج أخرى كثيرة منشورة يتضمن كل منها خمس أنشطة محورية ترتبط بالتحليل، التصميم، الإنتاج، الاختبار والمراجعة، الصيانة والإدارة.

ويناقش هذا الفصل أيضا كل من: النمذجة التمهيدية السريعة Rapid Prototyping فى هندسة البرمجيات وارتباطها بهندسة برمجيات المقررات التعليمية، أى تأثير برمجيات الوسائط المتعددة التعليمية على عملية التطوير، ووصف أنواع أدوات البرمجيات المختلفة التى يمكن أن تستخدم فى هندسة برمجيات المقررات التعليمية باستخدام التشابه والتناظر مع أدوات هندسة البرمجيات بمساعدة الكمبيوتر CASE فى هندسة البرمجيات، حيث يقود ذلك إلى اعتبار التقدم النهائى نحو آلية تطوير برمجيات المقررات التعليمية.

2 - التصميم التعليمى:

يتمثل الغرض من برمجيات التعليم أو التدريب فى دعم ومساندة التعلم، ويختلف ذلك إلى حد كبير من وظيفة برمجيات الأعمال كحزم الحسابات أو نظم دعم القرار الأكثر صعوبة. ومع برمجيات المقررات التعليمية أو مقررات برامج التدريب يتطلب تفسير قدرات المتعلمين البشرية، على ذلك يجب أن يتضمن التحليل والتصميم عمليات التعلم للطلاب المستهدفين التى لا تكون مفهومة بالكامل، وتختلف فى التفصيل من شخص لآخر.

وتعتبر عملية التعلم صعبة جدا من حيث تكرارها أو تقليدها بالإضافة إلى عدم إمكانية تطويرها بشمولية فى نموذج آلى (Barron et al, 1995).

ويطلق على تطوير التعليم للحاسبات الآلية والوسائل الأخرى "تصميم التعليم Instruction Design". وكلمة أو مصطلح "تصميم Design" تعنى فى الواقع

"تطوير Development"؛ حيث أنها تتضمن اعتبارات ومزاولات التحليل والتصميم والإمدادات السابقة مثل الوسائل والتقييم النهائي، وتتضمن كلمة "التعليم Instruction" عرض معد مخطط ذات طابع تعليمى تربوى، الذى قد لا يكون ملائما فى جميع المواقف التعليمية المختلفة والمتنوعة. ويرتبط ذلك بالتداخل التعليمى الذى يعنى أى نشاط من قبل المعلم لتشجيع عملية التعلم للطلاب. على أى حال، فإن مصطلح " التصميم التعليمي" يعتبر أكثر قبولا من "تطوير التداخل".

ويمثل التصميم التعليمى مدخلا منظما لتصميم التعليم والمواد التعليمية لتحقيق أهداف تعلم معينة، ويتعارض هذا المدخل مع الطرق التقليدية المستخدمة فى التعليم مثل تقليد المعلم كليا. والتصميم التعليمى يعتبر مدخلا مستقلا عن استخدام الحاسبات الآلية لإتاحة عملية التعليم وإمدادها. آراء "جاجنيه Gagne,1985" وزملاؤه من التربويين توضح جيدا أهمية التصميم التعليمى فى تأكيد نظرية التعلم. وصنف "جاجنيه" أنواع مخرجات التعلم محددا أن الطريقة المناسبة لتعريف هذه الأنواع تتمثل فى التساؤل عن كيف يمكن أن يعرض التعلم كمهارات عقلية ترتبط بما يلى:

- المفاهيم المعروضة بواسطة تعليم الأشياء وتصنيفها.
- القواعد المطبقة والمبادئ المعروضة.
- حل المشكلات التى تسمح بإنتاج وخلق الحلول والإجراءات.
- الاستراتيجيات المعرفية المستخدمة للتعلم.
- المعلومات اللفظية المحددة، المهارات الحركية التى تساعد الأداء الطبيعى.
- الاتجاهات المعروضة بواسطة تفصيل الخيارات المتاحة.

هذه المخرجات تمثل نتائج عمليات التعلم التى ترتبط بالمتعلمين الأفراد، وتقدم المهارات المحسنة المستهدفة من قبل المعلمين، وتعتبر شروط أو أوضاع التعلم

الخارجية (مثل التعليم أو التدريب) المؤثرة على التعلم مختلفة عن أنواع مخرجات التعلم المتعددة والمتنوعة. على سبيل المثال، قد يحتاج لعمل أشياء مختلفة لتعلم الاتجاهات بدلا من تعلم المهارات القلية أو الحركية. وعلى ذلك، يقترح "جاجنيه" (Gagne, 1985) بأنه على الرغم من اختلاف درجة التفصيل المطلوبة، فإن نفس أنواع النشاط التعليمى يحتاج إليها لكل عمليات التعلم ومخرجاته. كما يقترح "جاجنيه" أيضا وجود تسعة أحداث أو مواقف تعليمية عامة تعتبر متوافقة فى درجة التفصيل، على الرغم من أنها تتنوع مع نوع مخرج التعلم المحقق، ومع قناعة ورضاء التعلم المعين كما هو مبين فى الشكل التالى:

الحدث/ الموقف التعليمى الخارجى لعملية التعلم الداخلية
• جذب الانتباه: لتأكيد إدراك التعليم الوارد بحث ويحفز المتعلم.
• إخبار المتعلمين بهدف التعلم: إعلام المتعلمين ما سوف يقدرون عليه من التعليم المقدم.
• تحفيز وحث استدعاء التعلم السابق: من خلال التساؤل عن استدعاء المعرفة السابقة.
• عرض المثير الخاص بالحس على التعلم: عرض المحتوى المعرفى.
• إرشاد التعلم: المساعدة فى فهم (المعنى المكود) من خلال تقديم التنظيم والتوافق.
• استنباط الأداء المطلوب والمستهدف: سؤال المتعلم للاستجابة وعرض التعلم.
• تقديم التغذية العكسية: إعطاء تغذية عكسية عن أداء المتعلم.
• تقويم الأداء: يتطلب أداء أكبر للمتعلم ويقدم تغذية عكسية لتقوية التعلم.
• تعزيز حفظ سياق التعلم ونقله للآخرين: تقديم مزاولة هندسة برمجيات المقرر التعليمى.

شكل رقم (1/8): الأحداث / المواقف التعليمية

ويقدم ذلك نقطة بداية جيدة لتصميم أى موقف تعليمى، ويساهم فى اعتبار كيف يمكن الوصول إلى الأحداث التعليمية السابقة. وبذلك يمكن استخدام مخرجات التعلم إلى جانب الأوضاع الداخلية والخارجية المرتبطة بالتعلم، كما يمكن اختصار عملية التطوير التى يوصفها "جاجنيه Gagne" المتضمنة فيما يلى:

أولا: تحليل المتطلبات التعليمية من خلال هدف التعلم المستهدف:

1. تعريف أنواع مخرجات التعلم المرغوب تحقيقها.
2. تجزيء المخرج التى تعتبر صعبة ومعقدة فى تسلسل هرمى ضمن مخرجات التعلم والمتطلبات السابقة المعتمد عليها لإعطاء تسلسل هرمية التعلم مخرجات بسيطة.
3. تعريف الأوضاع الداخلية التى تحدث لتحقيق المخرجات للمتعلم.
4. تحديد الأوضاع الخارجية أو التعليم الذى يجب حدوثه لتحقيق الأوضاع الداخلية.

ثانيا: اختيار وسيلة التعلم:

1. تسجيل سياق التعلم.
2. تسجيل خصائص أو سمات المتعلمين.
3. اختيار الوسيلة للتعليم، من خلال الإجابة على السؤال التالى: كيف يمكن إتاحة الأحداث التعليمية؟ من خلال الكتب، السبورة البيضاء White Board، التعليم بمساعدة الكمبيوتر CAI، الفيديو... الخ.

ثالثا: تصميم التعليم وتخطيط الأحداث التعليمية لمساندة أنشطة التعلم:

1. التخطيط لمساندة المتعلم بواسطة استخدام أساليب الحث والتحفيز/ إتقان المهمة، أو الإنجازات.
2. لكل من مخرجات التعلم المخططة فى هرمية التعلم، تصمم الأحداث التعليمية السابق الإشارة إليها فى الشكل رقم (1/8) حتى يمكن أن تتطابق مع نوع مخرجات التعلم المطلوب فيما يتعلق بالمتطلبات المسبقة فى هرمية التعلم وبوسائل ملائمة واستخدامات المعلمين.

3. على الرغم من أن التعليم قد يظهر بأنه جاهز للاستخدام فى المزاولات الفعلية، فإنه يجب أن يختبر مسبق فى محاولات عديدة مع المتعلمين من خلال التقييم التوليدى / التكوينى Formative Evaluation .

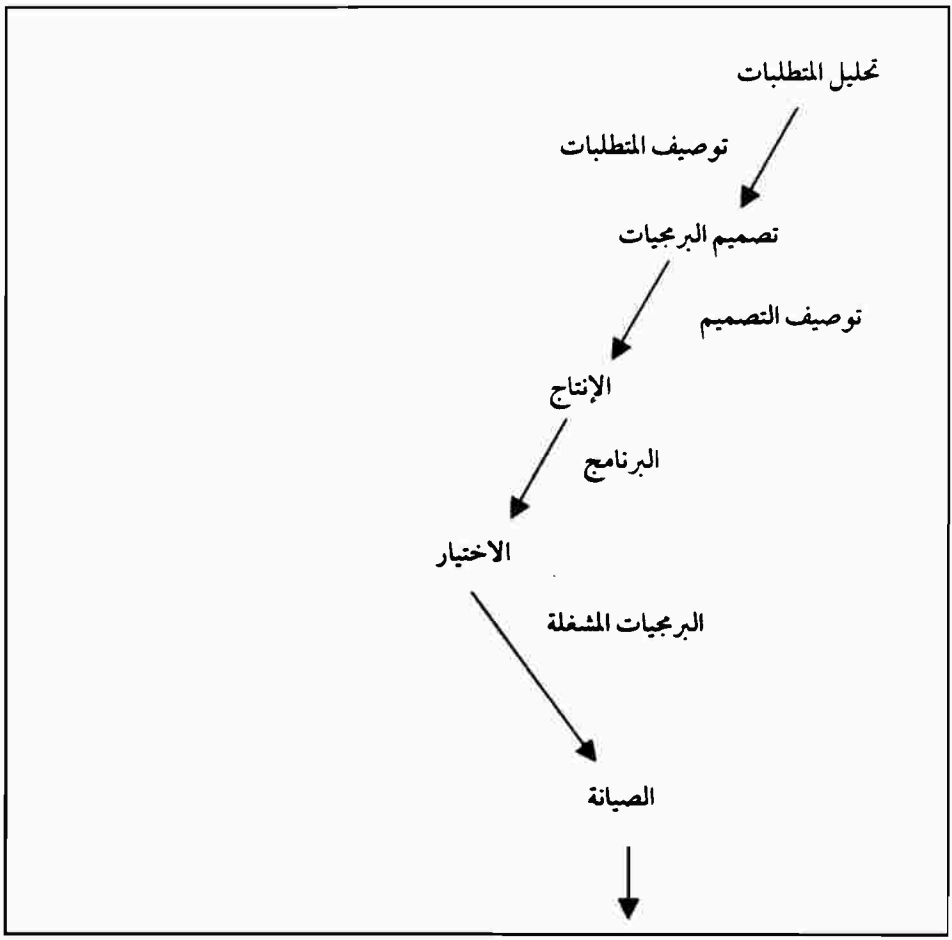
4. بعد استخدام التعليم، يمكن الحكم على فعاليته من خلال التقييم الجمعى Summative Evaluation .

مما سبق يمكن تحديد أن التصميم التعليمى "لجاجنيه" يؤدى إلى تحليل التعلم لكى ينجز ثم يترجم فى تصميم ملائم للأحداث التعليمية بطريقة فورية تساند عمليات المتعلم الداخلية، بعدئذ تختبر وتستخدم وتقيم، إلا أن "جاجنيه" لم يتعرض فى وصفه التصميم التعليمى إلى خلق أو إنتاج المواد التعليمية ذاتها التى تعرض لها كل من "بيترى وريجلوث (Petry, Mouton and Reigluth, C. M., 1987).

3 - هندسة البرمجيات:

بعد استعراض التصميم التعليمى يمكن التحول إلى مكونات هندسة البرمجيات Software Engineering من حيث البرمجة الحرفية و المنظمة من جهة، ومن خلال علم الحاسب الآلى الذى يبنى على أساس نظرى من جهة أخرى. وبذلك تعتبر هندسة البرمجيات عملية تفرض نظاما معينا على الأداء، وحتى يمكن إنتاج برمجيات الأعمال Business Software يمكن أن يحدد تحليل النظام اليدوى الحالى أو الحاجة لنظام جديد ما يحتاج لأدائه أو عمله. ويتمثل ذلك فى أن التصميم الذى يحدد كيف يمكن إنجاز النظام الكمبيوترى المستهدف حتى يمكن للمبرمجين تنفيذ هذا التصميم لإنتاج البرمجيات المطلوبة، والقيام باختبار ملائم لها حتى يمكن تركيبها ووضعها موضع التشغيل لأداء الوظيفة المطلوبة بطريقة آلية. أى أن هندسة البرمجيات تعنى مدخلا منظما لتطوير وتشغيل وصيانة وإدارة البرمجيات حتى يتوقف استخدامها فيما بعد لإعادة تطويرها بمتطلبات وتكنولوجيا جديدة (Van Vliet, 1993)

ويوضح الشكل التالى دورة حياة برمجيات Waterfall الخطية:



شكل رقم (2/8): دورة حياة برمجيات Waterfall الخطية

يتضمن الشكل السابق نموذج دورة حياة تطوير برمجيات Waterfall الخطية وفقا لما يلى:

- النشاط: البرمجيات المنتجة.
- تحليل المتطلبات توصيف المتطلبات.
- تصميم البرمجيات: توصيف التصميم.

- الإنتاج: البرنامج.
- الاختبار: البرمجيات المشغلة.
- الصيانة.

كما يوضح الشكل السابق تتابع الأنشطة بطريقة خطية Linear Sequence التى تمثل الطابع التقليدى لدورة حياة تطوير البرمجيات المشهور بنموذج Waterfall. وقد استخدم كثير من المطورين والكتاب مسميات وألفاظا مختلفة لوصف نفس الآراء ، إلا أنهم اتفقوا أساسا على التتابع الخطى المتسم به هذا المخطط أو النموذج. (Sommerville, 1989) ، (Van Vliet, 1993:).

مما سبق يتضح أن هذا النموذج يرتبط بالأنشطة أو المراحل الأساسية التالية:

- التحليل.
- التصميم.
- الإنتاج.

وتقيم دراسة الجدوى Feasibility Study ما إن الحل اقتصادى وعملى وفنى أم لا؟، وتعتبر هذه الدراسة تحليل تمهيدى لإنتاج خيارات التنفيذ المختلفة بتكاليفها وجدولة مدتها الزمنية.

وينتج تحليل المتطلبات وصف المشكلة التى ترتبط بالبرمجيات، ويتضمن ذلك وصف وظائف البرمجيات المحتاج لها، إمكانية توسعاتها فى المستقبل، التوثيق المحتاج إليه، وأداء المتطلبات مثل وقت الاستجابة، إلى جانب تضمين البيئة التى يشغل فيها هذا الحل (الأجهزة، البرمجيات، الاتصالات، المنظمة، المستخدمون). وناتج مرحلة التحليل يتمثل فى وثيقة توصيف المتطلبات التى تصف ما يحتاج إليه.

أما مرحلة التصميم فإنها تنتج نموذج النظام الكمبيوترى المطلوب لتلبية المتطلبات، وتجزأ الوظائف المطلوبة فى موديلات Modules وتفاعلاتها. بعدئذ تصمم واجهة التفاعل البينية مع المستخدم User Interface Design ، كما تحدد

هياكل البيانات Data Structure ، وتصميم العمليات التى تحول أنشطة التحليل إلى توصيف التصميم، إلا أن ذلك لا يرتبط بتفاصيل التنفيذ أو الإنتاج.

ويتضمن الإنتاج الذى يطلق عليه أيضا التكويد، التشفير أو التنفيذ خلق أو إنتاج البرمجيات التى يمكن تشغيلها واستخدامها. وتعتمد تفاصيل الإنتاج على إنشاء الأداء الذى يستخدم، إلا أنه توجد بعض المبادئ العامة التى يجب مراعاتها. كما قد توجد مرحلة انتقالية Transition يحول فيها توصيف التصميم المنطقى إلى توصيف أكثر تفصيلا، مثل استخدام لغة برمجة ذات مستوى عال High Level Language (HLL) للعمليات. ويتقدم الإنتاج من موديول إلى موديول آخر، ثم جمع كل الموديولات فى برمجيات عاملة قابلة للتشغيل.

وللاختبار أوجه عديدة ولا يحدث فقط بعد مرحلة الإنتاج، ولكنه يتم خلال كل مراحل عملية التطوير. ويحدث اختبار المستوى المتدنى Low Level وتصحيح الأخطاء Debugging عند كتابة كل موديول، كما قد يكون التآلف أو التعديل Tuning والتعظيم Optimization ضروريا بمجرد تجميع الموديولات معا. ويرتبط الاختبار بمراجعة أو تدقيق Verification منتج عملية التكويد أو التشفير بعد ترجمة صحة Validation توصيف البرامج بنفس النهج فى توصيف التصميم الذى يحقق ويراجع فى مواجهة توصيف المتطلبات، وبذلك تختبر الصحة Validation بأن منتج البرمجيات ما زال يحقق متطلبات المستخدم.

ويضع التركيب أو الإنشاء Installation البرمجيات المطورة فى عملية التشغيل الفعلى لها. وتوجد طرق مختلفة يمكن أن يؤدى بها ذلك.

أما الصيانة Maintenance فإنها توضح أن البرمجيات قد تشتمل على أخطاء غير متنبأ بها فى البداية، بالإضافة إلى أن هذه المرحلة سوف تتطلب إلى تكيف وتحسين مستمر عبر حياتها. وفى هذا الصدد يمكن ملاحظة أن صيانة البرمجيات التجارية قد تكلف نصف تكلفة التطوير الكلى.

يمثل العرض السابق نظرة بسيطة لعملية تطوير البرمجيات. وتنتج كل مرحلة شيئًا ما قد يكون خططا، رسومات وأشكال، أو شفرات... الخ. وبصفة عملية يؤدى هذا المدخل المنطقى التتابعى من خلال عمليات التغذية العكسية، ويرجع السبب فى ذلك إلى أن التطوير فى مرحلة تالية لا يمكن أن يتحقق إن لم تكن المرحلة السابقة صحيحة كليا، وعلى ذلك يجب القيام بعمليات المراجعة والتدقيق بصفة مستمرة، مما يعنى جدوى تطبيق الطريقة التتابعية الخطية.

ويمكن تلخيص العرض السابق بأن هندسة البرمجيات تمثل مدخلا منظما لتطوير البرمجيات، وأن نموذج Waterfall هو نموذج تتابعى خطى مع تغذية عكسية لأنشطة التحليل والتصميم والإنتاج التى تتبع بواسطة الاختبار والصيانة.

4 - هندسة برمجيات المقررات التعليمية:

تمثل هندسة مقررات البرمجيات التعليمية Courseware Engineering مجموعة من المزاولات، الأدوات والمنهجيات التى تنتج من محاولات تطبيق المدخل الهندسى لإنتاج برمجيات المقررات التعليمية، كما تعتبر المدخل الهندسى الذى يختلف عن المدخل الحرفى Craft Approach، حيث يركز ويؤكد استخدام الطرق والأدوات المعيارية المقننة بدلا من الحدس والتخمين، إلى جانب تقدير قيمة نقل العمليات والنتائج بدلا من الخلق المتسم بالخصوصية المطلقة. (Goodyear, 1995)

ويرتبط تطوير برمجيات المقررات التعليمية بكل من هندسة البرمجيات العامة والتصميم التعليمى للتداخلات التعليمية، كما يمثل مدخلا ممكن التطبيق، وبذلك يعتبر جزءا من هندسة البرمجيات التى تعالج بأسلوب خاص بها. وعلى هذا الأساس، فإن تركيز هذا المدخل وتأكيده بصورة واضحة لا لبس فيها، يرتبط بالتعلم البشرى واكتساب المعرفة ووضع برمجيات المقررات التعليمية فى مجموعة مستقلة خاصة بها. (De Diana and Van Schaik, 1993)

وتطوير المواد التعليمية للتدريس والتعلم التى تحدث بدون توظيف استخدام

الحاسب الآلى تتطلب نفس مرحلتى التحليل والتصميم السابق الإشارة إليهما، إلا أن مراحل الإنتاج والاختبار والصيانة فإنها تعتبر مراحل مختلفة وخاصة عندما تكون الوسيلة أو الوسيط التعليمى مواد مطبوعة أو توظف الحاسب الآلى. على سبيل المثال، فإنه لتطوير برمجيات مقررات تعليمية متماثلة أو متطابقة مع هندسة البرمجيات، فإن ذلك يرتبط على وجه الخصوص فيما يتصل ببعض اوجه التصميم كما فى حالة واجهة التفاعل البينية مع المستخدم، والإنتاج المرتبط بالتكويد أو التشفير؛ حيث إن الوسيلة هى نفسها. وإن أدوات الإنتاج أى لغات البرمجة قد تكون متشابهة إلى حد كبير، إلا أن المراحل الأولى فى عملية التطوير تعتبر مختلفة تماما. وبذلك تعتبر هندسة برمجيات المقررات التعليمية شبيهة إلى حد ما مع حرفية Grafting مراحل التطوير التعليمى الأولى التى ترتبط بمراحل تطوير البرمجيات التالية لها حتى تعطى طريقة التطوير اللازمة لتطوير برمجيات المقررات التعليمية.

ويصف كلا من "دين وويتلوك (Dean and Whitlock, 1991) عملية التطوير الأساسية فى بيئة تجارية ترتبط بالتدريب كما يلى:

- استقصاء المشكلة والتعرف عليها (أى التحليل).
- تخطيط المقرر التعليمى أو التدريبى (أى التصميم التعليمى).
- التطوير (أى الإنتاج).
- التنفيذ والتقييم (أى التركيب والتشغيل والتقييم).

والعرض التالى يفصل الأوجه السابقة:

أولا: استقصاء المشكلة (التحليل):

1. تحليل الغرض من التطوير وتعريف الأداء المطلوب.
2. تحليل حاجات التدريب (تحليل الواجهة النهائية للتدريب) مع تعريف القصور فى الأداء وأسبابه.
3. إنشاء الحاجة للتدريب باستخدام الحاسب الآلى.

ثانيا: التخطيط (التصميم).

1. وصف المهام المطلوب تعلمها أو التدريب عليها.
2. تجزيء هرمى للمهام فى مهام فرعية تنبثق منها.
3. تعريف سمات جمهور المتدربين أو المتعلمين المستهدف.
4. تحديد وحدات المقرر ولكل درس يحدد الموديول Module الخاص به.
5. تحديد المهارات المفصلة المحتاج إليها والمحتوى المفصل وتتابعه.

ثالثا: التطوير (التصميم المفصل والإنتاج):

1. تفسير المحتوى التعليمى كمجموعة قواعد محددة (التصميم المفصل).
2. تقرير خطوات التعلم (التصميم التعليمى)، تقسيم مجموعة القواعد فى مقادير Chunks لتقرير حجم الدرس قبل الاختبار.
3. تقرير تتابع الشاشات المرتبط بخرائط التدفق Flowcharts (التصميم)
4. تصميم الشاشات كلوحات عرض Storyboards (التصميم).
5. إنتاج التكويد أو التشفير فى لغة برمجة أو لغة تأليف (الإنتاج).

رابعا: التنفيذ والتقييم (الاختبار):

1. تقييم الزملاء قبل الاستخدام.
2. تصحيح المقرر مع مجموعة تجريبية من المتعلمين أو المتدربين.

ويتم مخطط التطوير السابق الإشارة إليه نحو التعليم بمساعدة الكمبيوتر CAI مع النص والرسومات، ويؤكد ذلك أن البرمجة التى تمثل جزءا صغيرا من دورة حياة تطوير البرمجيات تعمل بواسطة المبرمجين الذين يستخدمون لغة التأليف Authoring Language أو لغات الغرض العام General Purpose Languages.

وقدم كلا من "استيفن أليس ، استانلى ترولیب" (Allessi and Trollip, 1991)

فى كتابهما "التعليم المبنى على الكمبيوتر" مخطط تطوير برمجيات مقررات تعليمية أكثر تفصيلا، كما أنهما اهتما بصفة خاصة بالبرمجيات التعليمية التى تشتمل على التدريب والمحاكاة والتمارين، إلا أنهما يخاطبان المدرسين العاملين فى مجموعات صغيرة. ويشتمل مخططهما أو نموذجهما على عشرة خطوات أساسية ترتبط بتطوير أحد الدروس الفردية، أما الأدوات المستخدمة فسوف يتعرض لها هذا العمل فى بند لاحق. وتتمثل هذه الخطوات فى التالى:

(1) **تقرير الحاجات والأغراض (التحليل)**: ويعنى ذلك وصف خصائص المتعلمين قبل بدء عملية التعليم وتحديد القدرات الجديدة بعدئذ.

(2) **جمع الموارد (التصميم)**: ويتضمن ذلك المادة المرتبطة بالموضوع، الموارد المختلفة للتصميم التعليمى (لوحات العرض Storyboard، أدوات البرمجيات، الموارد البشرية) بالإضافة للموارد المحتاج إليها للإمداد أو الإتاحة (الكمبيوتر، الأسلوب اليدوى، الخبرة).

(3) **تعلم المحتوى (التحليل)**: يجب أن يتعلم أو يلم المطور بالمحتوى حتى إذا عمل مع شخص أو أشخاص آخرين خبراء أو متخصصين فى هذا المحتوى، كما سوف يتعلم أيضا خبير المجال الموضوعى التصميم التعليمى وأبعاده المختلفة. وسوف يكون منتج هذا التعلم عبارة عن عروض عن الموضوع تأخذ شكل شبكة دلالية Semantic Network ، هرمية المفاهيم Concepts Hierarchy ، خرائط الموضوع Topic Maps، أو خرائط تدفق الإجراءات بالاعتماد على ما إن كان المحتوى يتمثل فى مهارات تعليمية، معلومات لفظية، استراتيجيات معرفية، اتجاهات أو مهارات حركية.

(4) **خلق أو توليد الأفكار (التصميم)**: تستخدم طريقة العصف الذهنى Brainstorming لخلق أو توليد الأفكار الجديدة للمحتوى الذى سوف يدرس وطرق تدريس التى تستخدم.

(5) **التصميم التعليمى (التصميم)**: ويشتمل ذلك على التالى:

- اختيار الأفكار الأحسن.
- أداء تحليل المهمة.
- أداء تحليل المفهوم عن المحتوى.
- عمل خريطة تعلم.

(6)**خرائط تدفق الدرس (التصميم)**: تستخدم خرائط تدفق Flowcharts الدرس لتقرير تتابع **المادة التعليمية المقدمة فى الدرس**.

(7)**لوحات العرض (التصميم)**: ترتبط لوحات العرض Storyboards بمحتوى المخرج التعليمى المفصل الموجه للمتعلم، ويصمم ذلك على الورق.

(8)**برمجة الدرس (الإنتاج)**: ويرتبط ذلك بإنتاج البرمجيات العاملة التى تستخدم لغة البرمجة، أو نظام التأليف أو الأداة المعينة.

(9)**إنتاج المواد المساندة (الإنتاج)**: ترتبط المواد المساندة للتعلم المنتجة فى أدلة الطلاب، أدلة المعلمين، الأدلة الفنية، ملاحق التعليم... الخ.

(10) **التقييم والمراجعة (الاختبار)**: ويتمثل ذلك فى: التقييم قبل الاستخدام بواسطة مراجعة الزملاء، وتقييم الاستخدام ومخرجات التعلم مع متعلمين فعليين فى اختبار تجريبى معين.

مما تقدم يلاحظ أن "استيفن أليس ، استانلى ترووليب" قد حددا نقاطًا عديدةً ترتبط بنموذجهما والذى تبنياه التى يمكن إجماله فى التالى:

1. يأخذ التقييم والمراجعة مكانا فى نقاط عديدة ولا يقتصر فقط على أداء ذلك عند نهاية التطوير.
2. البناء على مبادئ علم النفس المعرفى **Cognitive Psychology** المرتبط بالإدراك، الفهم الانتباه، الذاكرة، التعلم النشط، الحث، موقع الرقابة، نقل التعلم والاختلافات الفردية.
3. يعتبر الابتكار والإبداع عاملا رئيسيا للتصميم الجيد.

4. تتابع المناقشة من التصميم الورقى إلى تنفيذ البرمجيات، حيث يجب تأجيل استخدام الكمبيوتر.

5. يعتبر مدخل فريق العمل هو المدخل الأفضل، حيث يشتمل ذلك عل أفكار مبتكرة جديدة، كما يمكن أن يكون ذات طبيعة أكثر ذاتيا فى النقد.

ويمكن تلخيص ما تقدم فى أن هندسة برمجيات المقررات التعليمية هى هندسة البرمجيات بصفة عامة التى تطبق على المقررات أو البرامج التعليمية، ولكن يعنى المتطلب للتعلم تواجد أنشطة أكثر ذات توجه مختلف عن برمجيات الأعمال. وتعتبر مرحلتا التحليل والتصميم من التصميم التعليمى التى تأتى فيها مرحلتا الإنتاج والاختبار من هندسة البرمجيات، وتتضمن المخططات العملية أساليب تحليل وتصميم عديدة.

5 - نماذج هندسة برمجيات المقررات التعليمية:

على الرغم من مناقشة نموذجى تطوير برمجيات المقررات التعليمية لكل من "ديـن ووايتلـوك Dean and Whitlock " و " أليـس وتـروليـب Alessi and Trolip "السابق الإشارة إليهما، إلا أنه توجد نماذج عديدة لتطوير برمجيات المقررات التعليمية منشورة ومتاحة للباحثين.

وبدراسة نماذج التطوير المتاحة يمكن ملاحظة أنها تختلف عن بعضها البعض لا بسبب نظريات التعليم الظاهرية التى توضحها، ولكنها تختلف أكثر بسبب تطبيق طرق عمل مختلفة فى ظروف مختلفة. ويمكن تجميع الخطوات المختلفة التى تتضمنها هذه النماذج العديدة فى المجموعات الوظيفية التالية:

1. التحليل.
2. التصميم.
3. الإنتاج.
4. التقييم والمراجعة.

5. إدارة العمليات.

وتعتبر الوظائف الثلاثة الأولى ذات طبيعة عملية ووظيفية بحتة متطلبة لإنتاج البرمجيات. فقبل الإنتاج يجب تصميم وظائف البرمجيات، وقبل ذلك أيضا يجب فهم المتطلبات التى يجب أن يرضى عنها المستخدمون، أى يجب فهم المتعلم وأهداف التعلم والموضوع المرتبط بذلك.

ويطبق التقييم التوليدى والمراجعة فى موقف أو أكثر من موقف فى العملية التعليمية، حيث يطبق دائما بعد مرحلة الإنتاج قبل الإنشاء والاستخدام وفيما بعد ذلك فى بعض الأحيان، كما يطبق أيضا على منتجات مرحلتى التحليل والتصميم. وبصفة عامة، كلما كثرت دوائر التقييم والمراجعة فإنه سوف تتوافر منتجات أحسن. ويمكن أن ينتج فى مراجعة منتج المرحلة أو النشاط السابق أى الرجوع فى دورة حياة Waterfall الخطية (Van Vliet, 1993:33). وتختلف طرق التقييم الدقيقة اعتمادا على مرحلة التطوير وحجم المشروع. ويمكن عرض الأفكار المبدئية فى حلقة نقاش أو مختبر يختص بموضوع التفتيش، كما يمكن إخضاع التصميمات والتكويد أو الشفرة للمتابعة المستمرة Walkthrough، وأيضا تختبر البرمجيات المشغلة أو العاملة للتحقق من إمكانية الاستخدام Usability ومراجعات الزملاء والاختبارات الميدانية. وتتضمن النمذجة التمهيدية Prototyping تقييم ومتابعة البرمجيات العاملة.

ويقرر حجم المشروع أهمية عمليات الإدارة التى تصبح ضرورية بمجرد تطوير برمجيات المقرر التعليمى على أساس مختلف من خلال المعلم مثلا لطلابه. وإلى حد كبير، يحتاج إلى فريق عمل لإدارة المشروع. وتشتمل الاعتبارات التجارية على القضايا المالية والتعاقدية. على سبيل المثال، تحديد خطوات التطوير الأساسية، تقدير التكاليف والارتباط بالميزانيات، ويوجد توجهان نحو ذلك: جدول الوقت والموارد المحتاج إليها. والبدء بالمتطلب التمهيدى من العميل الذى يطلب تطوير البرمجيات. وجدول الوقت المقدر لكل مرحلة وعملية يجب أن يدار عند التزام

المشروع بالوقت والميزانية. وتتطلب إدارة الوقت منتجات أو خطوات تنفيذ فى العملية، كما يتضمن تحديد الموارد تجميعها وتخصيصها إلى جانب إدارة الابتكار فى نطاق عملية العصف الذهنى.

ويصف "فوستر" (Foster, 1993) إدارة مشروعات برمجيات المقررات التعليميـة، كما يصف فعالية التكلفة "فريدلر وشابو" (Friedler and Shabo, 1991).

وعلى ذلك يمكن طرح السؤال التالى: لماذا نحتاج لطريقة تطوير برمجيات المقررات التعليمية؟، ويمكن الإجابة على هذا السؤال فيما يتصل بالعرض التالى:

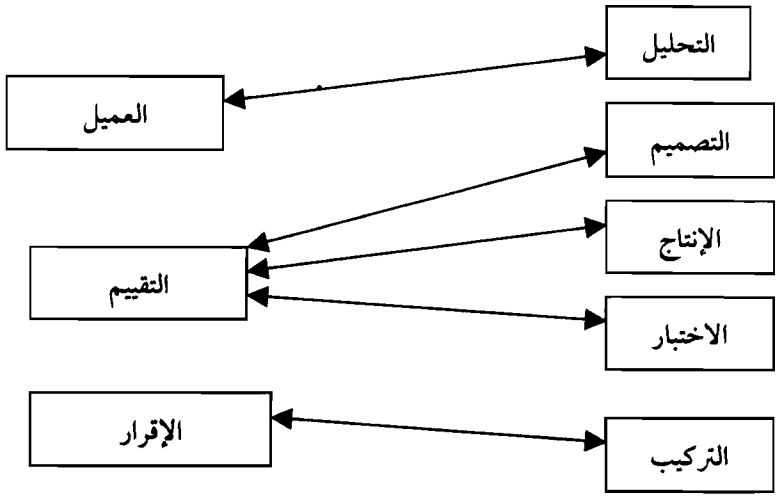
(1) مستويات التجريد المختلفة التى يحتاج إليها المطورون:

تعتبر عبارة المتطلبات التمهيدية مستوى من مستويات التجريد Abstraction المتعلقة بالمشكلة المحتاج إلى حلها، وفى كل مرحلة من مراحل دورة حياة التطوير التتابعى أو الخطى تنتج حلول محددة وأكثر تفصيلا، وتنتهى بحل برمجيات معينة. وحتى إذا لم تنتج تابعيا؛ فإن المستويات المختلفة التى يحتاج إليها يجب توافقها معا.

(2) الإدارة والرقابة:

يحتاج مديرو التطوير القيام بخطوات معينة فى توقيتات محددة وعند نهاية خطوات تنفيذ المرحلة المعينة فيما يتصل بالمخرجات Deliverables حتى يستطيعوا مراجعة ومراقبة تقدم عملية التطوير.

ويجب ألا تستبعد تقييمات نموذج Waterfall الخطية بواسطة العميل. ولكل مرحلة منتجات مثل وصفات الحاجات المحللة نصيا ورسوميا، والوظائف المصممة، وأخيرا سمات أو خصائص البرمجيات. ويمكن تقييم هذه المنتجات كما فى الشكل التالى المرتبط ببيئة تجارية سوف يتعاقد عليها:



شكل رقم (3/8): تقييم منتجات تطوير البرمجيات

يوضح الشكل السابق المراحل أو الأنشطة الرئيسية التالية ودور التقييم والإقرار فيها:

- التحليل ------ تقييم وإقرار العميل.
- التصميم ------ تقييم وإقرار العميل
- الإنتاج ------- تقييم وإقرار العميل
- الاختبار ------ تقييم وإقرار العميل
- التركيب ------ تقييم وإقرار العميل

وباختصار توجد مخططات أو نماذج كثيرة منشورة لتطوير برمجيات المقررات التعليمية التى تعتبر تجمعات مختلفة من أنشطة التحليل، التصميم، الإنتاج، التقييم والمراجعة والإدارة.

6 - إعداد النمذجة التمهيدية:

فى تطوير البرمجيات، قد يحل محل دورة الحياة التقليدية الخطية Waterfall فى بعض الأحيان طريقة إعداد النمذجة التمهيدية السريعة Rapid Prototyping حيث

إن العملية المطلوب تحويلها آليا باستخدام الحاسب الآلى تفهم جيدا وأحسن، لأن توصيف المتطلبات التمهيدية يكون أدق، بينما عندما تكون المتطلبات غير واضحة أو مبهمة أو عندما يتغير وضعها، لأن الطريقة المستخدمة تعتبر غير مناسبة ويفضل استخدام طريقة دورة الحياة المتعاقبة والخطية بدلا منها.

وعند فحص مخاطر تطوير البرمجيات، فإن المخاطر الأعم تهمل الموارد ويبنى على أساسها نظاما خطأ.

وحتى عند توافر الموارد غير المحددة، فإن المطورين قد يبنون النظم التى لا تلبى احتياجات المستخدمين (Maude and Willis, 1991)، كما توجد فجوة كبيرة جدا بين المستخدمين الذين عبروا عن متطلباتهم والبرمجيات الجاهزة للاستخدام.

وبذلك بزغت طريقة دورة حياة النمذجة التمهيدية Prototyping Life Cycle التى تحتم دقة وتوافق المتطلبات واكتمالها وواقعيتها عندما تكون أساس تطوير البرمجيات. وأصبح ممكنا فحص هذه المتطلبات من قبل المستخدمين والمطورين معا بطريقة يدوية على الورق فى بداية الأمر، ثم بعدئذ باستخدام برمجيات لذلك حتى يتأكد المستخدمون أكثر على متطلباتهم. وتحقق طريقة إعداد النمذجة التمهيدية إنتاج البرمجيات النهائية بسرعة كبيرة وتعطى المستخديون انطباعا مرضيا وإيجابيا بما سوف تكون عليه هذه البرمجيات عند تطويرها نهائيا. (Sommerville, 1989) ، (Van Vliet, 1993) ، (Schack, 1990). وتمثل النمذجة التمهيدية نموذج سلوك نظام البرمجيات الممكن استخدامه لفهم النظام المقترح أو أوجه معينة خاصة به، كما يسهم فى توضيح المتطلبات بدقة . (Maude and Willis, 1991:50)

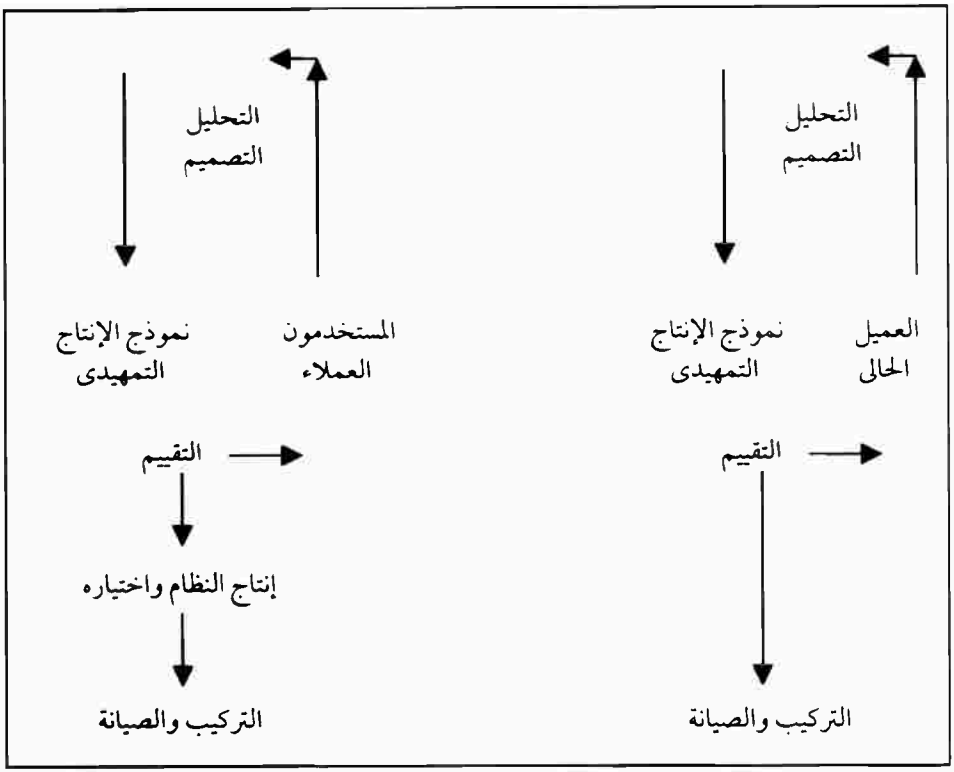
ويختلف النموذج التمهيدى من المنتج النهائى للبرمجيات عن طريق تقليل النواحى الوظيفية والتعقيد التى قد تؤثر على بطء وتدنى وضعف البرمجيات المطورة، وبذلك يعد للنموذج التمهيدى المدخلات الخطأ من قبل المستخدم، وعلى ذلك قد تستخدم لتوضيح مبادئ استخدام النظام المخطط وإمكانيته.

ويعتبر عمل إعداد النمذجة التمهيدية أسهل وممكن تنفيذه بسهولة بواسطة استخدام أدوات إنتاج البرمجيات مثل لغات الجيل الرابع 4th Generation Languages الأسرع فى الاستخدام من لغات البرمجة التقليدية، حيث تكتب البرامج كنص يجمع وينفذ بعدئذ.

وبدون استخدام أدوات البرمجة المتقدمة يصبح إنتاج النماذج التمهيدية للبرمجيات مكلف جدا مثل المنتجات النهائية التى أنتجت فى الماضى ولا تستخدم حاليا. وتتوافر أيضا أدوات البرمجة الحديثة لإنتاج برمجيات المقررات التعليمية مثل أدوات التأليف Authoring Tools مما يمكن إعداد النمذجة التمهيدية عند التطوير.

ويستخدم أحد المداخل فى هندسة البرمجيات النموذج التمهيدى كنموذج مرئى غير كامل لمساعدة المستخدم أو العميل فى رؤية المنتج وتحسين توصيف المتطلبات (ومع ذلك قد يبنى النموذج التمهيدى البرمجيات من لا شيء بطريقة عادية تستخدم لغة البرمجة، كما قد تحفظ بعض مكوناته المعينة كما فى حالة واجهة التفاعل مع المستخدم) ودمجها معا فى المنتج النهائى الذى سوف يكون أحسن بناءًا. ويحتاج تطوير البرمجيات التعليمية إلى سرعة التنفيذ.

من جهة أخرى، يمثل مدخل النمذجة التمهيدية المرتبط بالبرمجة الاستكشافية Exploratory Programming شكلا يعرض المستخدمين مع برمجيات غير مكتملة على الرغم من معرفتهم لها ومن خلال آرائهم يمكن تحسينها وتعزيزها. وبذلك يمكن إنجاز تطوير البرمجيات بسرعة حتى الوصول إلى الإصدارة المقبولة. ومن المحتمل حصول المستخدمين على ما يريدونه، إلا أن هناك احتمالاً برداءة البرمجيات فنيا. وقد يؤدى الهيكل الداخلى السيئ إلى برمجيات غير مصقولة أو مكتملة إضافة لصعوبة صيانتها وبطء إنجازها عند استخدام نظام تأليف لإنشائها كما هو مبين فى الشكل التالى:



شكل رقم (4/8): النمذجة التمهيدية

ويشتمل نموذج التطوير المتتابع الخطى Waterfall على بعض التغذية العكسية أو المرتدة من العملاء لنشاط أو أكثر أو حتى لكل الخطوات، أما نموذج النمذجة التمهيدية فإنه يعتبر امتدادًا للنموذج الخطي، حيث إن المنتجات الجزئية تطور بصفة مكررة لتوضيح نشاطى التحليل والتصميم.

وعلى ذلك يمكن التساؤل: هل يجب استخدام النمذجة التمهيدية عند تطوير برمجيات المقررات التعليمية؟، والإجابة على ذلك تكون بالإيجاب.

ويمكن أن تساند برمجيات المقررات التعليمية أدوات النمذجة التمهيدية الوظيفية Functional Prototyping وأسلوب هندسة البرمجيات بمساعدة الكمبيوتر CASE بطريقة تتسم بالاحتمالية لإضفاء كفاءة تطويرها (De Diane and Van Schaik, 1993).

ويعتبر التعلم من الأعمال التى يصعب التنبؤ بها، فهو مختلف بالنسبة للأشخاص والأوضاع والمواقف المختلفة والمتنوعة. وعلى ذلك تصبح مرحلة التحليل للبرمجيات صعبة لحد كبير، كما أنه من غير المحتمل الحصول على تحليل وتصميم صحيح بالكامل من المرة الأولى، على الرغم من أن النمذجة التمهيدية تسهم فى تقييم الاستراتيجيات التعليمية فى مراحل تطوير النظم الأولى. وعلى أى حال، تتفق النمذجة التمهيدية بصفة جوهرية مع واجهات تفاعل المستخدم التى تكون مهمة جدا فى برمجيات المقررات التعليمية.

وفى هذا الصدد، توجد مشكلة خاصة مع النمذجة التمهيدية لبرمجيات المقررات التعليمية التى تتمثل فى أنه لا يمكن تقييم فعاليتها للتعلم. ففى برمجيات الأعمال Business Software قد يكون النموذج التمهيدى مجرد ترتيب الشاشات Screen Layout أو ترتيب العمل ولكنها تمثل إصدارات برمجيات تتسم بالبساطة. وقد يكون ذلك كافيا لدعوة المستخدمين لرؤية ما سوف تكون عليه البرمجيات النهائية من خلال تحسين توصيفها. إلا أنه مع برمجيات المقررات التعليمية لا تعرض النماذج التمهيدية على المتعلمين، كما أنها لا تعمل مطلقا فيما يختص بإنتاج أهداف التعلم. وفى الحقيقة يعتبر رأى المتعلم عن البرمجيات مهما ومفيدا جدا على الرغم من أن المتعلمين أو المستخدمين لا يستطيعون الحكم على ما إن كانوا سوف يستطيعوا التعلم بها أم لا. وبذلك يجب أن يرتبط هذا التقييم بالاختبار التجريبى والاختبار الميدانى اللاحق.

على أنه فى الغالب، تنتج برمجيات المقررات التعليمية للعميل المتمثل فى المدرس أو المدرسة أو الشركة بغية تدريب عامليها. وقد يستطيع هذا العميل الحكم على مدى فعالية النمذجة التمهيدية المتبعة فى تطوير البرمجيات التعليمية مما قد يساعد فى تحسين المتطلبات الخاصة بالمقرر التعليمى وتصميمه؛ حيث إن النمذجة التمهيدية توضح متطلبات العميل، كما أنها تعتبر أكثر أهمية فى ترضية العملاء من المتعلمين ذاتهم عما هو الحال من وجهة النظر التجارية.

وتعتبر النمذجة التمهيدية ذات قيمة وبصفة خاصة عند تحديد المتطلبات بوضوح كاف. على سبيل المثال، اقترح بعض الباحثين من أمثال "ساندفورد" (Sandford, 1990) وكل من "تريب و بيشلماير" (Bichelmeyer, 1990 Tripp and) استخدام النمذجة التمهيدية عند تطوير الوسائل أو الوسائط المتعددة التفاعلية، كما اقترح أيضا "فريدلر وشابو" (Friedler and Shaboo, 1991) عند استخدام نظم التأليف فى تطوير برمجيات المقررات التعليمية. وقد أوصى كل من "ويت و واجر" (Witt and Wager, 1994) بشىء شبيه بهذا التوجه عند تطوير نظم دعم الأداء الإلكترونية أثناء وقت التعليم أو التدريب.

ومن الملاحظ أنه من الممكن إساءة استخدام مصطلح "النمذجة التمهيدية" ليعنى تقييم العميل كل المنتجات بدلا من البرامج العاملة ذاتها مثل تصميم الوثائق، إلا أن ذلك يكون مجرد جزء من دورة حياة تطوير البرمجيات بطريقة خطية Waterfall التى تستخدم الأحداث الرئيسية Milestones كما سبق تحديده فى الشكل السابق (شكل رقم 3/8).

وبينما يكون من المفيد تضمين المستخدم فى كل مراحل تطوير البرمجيات كما فى دورة حياة الخطية فى تطوير النظم، فإن هندسة البرمجيات الحديثة تعتمد فى الأساس على محورية دور المستخدم ذاته فى طرق التطوير، وبذلك يصبح من الأجدر قصر مصطلح "النمذجة التمهيدية" على تطوير البرمجيات الحديثة.

ومن هذا المنطلق أسهمت أدوات التأليف Authoring Tools فى فعالية تكلفة Cost Effectiveness وكفاءة النمذجة التمهيدية المرتبطة بهندسة البرمجيات الحديثة. وبذلك أصبح ممكنا إنتاج البرمجيات الحديثة باستخدام لغات البرمجة ذات الغرض العام مثل (لغة البيزيك ولغة البيزيك المرئى V. Basic) ولغات التأليف المتخصصة مثل (لغات Tencore or PC-CAI) أو لغات التأليف المتخصصة التى تتطلب قليلا من نص الشفرة أو لا تتطلب ذلك مثل (لغات Authorware, Director or Optima) . ويساعد استخدام أدوات التأليف فى تجريد البرمجيات وزيادة إنتاجيتها وتعظيم إعادة استخدام مكونات البرمجيات بطريقة متزامنة. بالإضافة لذلك فإن

الانتقال الحديث لتوظيف النوافذ Windows وواجهات التفاعل الرسومية مع المستخدم Graphical User Interface (GUI) ساهم فى زيادة جهد البرمجة، وبالتالى القيمة المضافة لأدوات التأليف فى البيئات التعليمية.

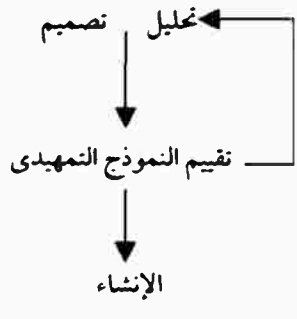
وفى هذا الصدد طورت شركة Sanderson CBT المتخصصة فى إنتاج برمجيات المقررات التعليمية فى الولايات المتحدة الأمريكية نموذجا لتطوير برمجياتها عند استخدامها لغة تأليف تقليدية (تنكور Tencore) كما هو موضح فى الشكل التالى رقم (5/8) الذى يبين أيضا النموذج الجديد الذى تتبناه الشركة بعد الانتقال لاستخدام بيئة النوافذ مع حزمة برمجيات Toolbook كأداة إنتاج للبرمجيات.

دورة الحياة التقليدية (التحليل)	دورة الحياة النمذجة التمهيدية (التحليل)
ـ طلب العميل	ـ طلب العميل
ـ التحليل المبدئى	ـ التحليل المبدئى
ـ النموذج التمهيدى على ورق	ـ النموذج التمهيدي على برنامج PP
ـ تحليل الحاجات والمتطلبات تفصيليا	ـ تصميم الحاجات والمتطلبات تفصيليًا
ـ التقرير للعميل	ـ التقرير للعميل (انتهاء التحليل)
(التصميم)	(التصميم والإنتاج)
ـ إطار لوحة التصميم	ـ اختيار إمكانية الاستخدام
ـ التصميم المبدئى	ـ تحديد دورة التعديل
ـ تصميم المحتوى الأول	ـ انتهاء التصميم
ـ اختيار إمكانية الاستخدام	ـ تصميم المحتوى الشامل فى Toolbook
ـ دورة التعديل	ـ التحليل والاختبار
ـ التقرير للعميل (إنتهاء التصميم)	ـ التفتيش على الجودة
ـ تصميم المحتوى الشامل	ـ الانتهاء من موديل بعد الآخر
الإنتاج	
ـ برمجة موديل بعد الآخر	

شكل (5/8): دورة حياة شركة Sanderson CBT باستخدام النمذجة التمهيدية

يلاحظ فى الشكل السابق اختفاء مرحلة البرمجة، وبذلك لا تطلب الاستعانة بمبرمجين متخصصين فى لغة Tencore. وبذلك يرتكز تطوير برمجيات المقررات التعليمية على مصمم التعليم أو التدريب وفنان الرسومات. وقد أوجد ذلك تحولا واضحا تجاه استخدام طريقة النمذجة التمهيدية (لا البرمجة الإكتشافية Exploratory Programming) التى ساهمت فى إمكانية استخدامها أدوات التأليف. وبدلا من النماذج التمهيدية على الورق أو على شاشات ثابتة وجدت طرق وحزم برمجيات مبتكرة مثل حزمة Toolbook التى تستخدم فى عرض النماذج بطريقة تتسم بالتفاعلية، وقد درب المصممون على استخدامها بكفاءة وفاعلية. وعلى الرغم من ذلك، يحتاج مشروع تطوير برمجيات المقررات إلى مبرمجين متخصصين فى (لغة سي C) لتثبيت البرمجيات الخاصة. وإلى جانب ميزة الإنتاجية التى تتسم بها النَّمذجة التمهيدية، فإن دورة الحياة المختصرة تعتبر أكثر مرونة من دورة الحياة التقليدية. ويتضمن تفتيش جودة البرمجيات اختبار إمكانية استخدامها Usability، حيث يكون أكثر ظاهريا قبل الانتهاء من البرنامج.

وعند تأمل تطوير البرمجيات على المدى البعيد، نلاحظ سهولة تطوير واستخدام أدوات إنتاج البرمجيات. ولن يكون من الضرورى التصميم أولا على الورق. وسوف تتلاشى المشكلات المتعلقة بتطوير النماذج التمهيدية المتعاقبة، عندئذ سوف يرتكز مع التحليل والتصميم فقط قبل الإنشاء ولا يستدعى ذلك الاستمرار فى التعاقب حيث تتم المراحل بالتوازى. وقد يلخص نموذج تطوير برمجيات المقررات الدراسية فى الشكل التالى:



شكل رقم (6/8): التحليل / التصميم ◄► التقييم

يلخص الاستعراض السابق أن النمذجة التمهيدية تقدم تقييم ومراجعة مبكرة عما يوفره نموذج دورة الحياة التقليدية Waterfall بتوفير برمجيات للعميل تساعد فى تقييم البرمجيات مبكرا، وقد يؤدى ذلك إلى مخاطر ترتبط بتوصيف المتطلبات بدقة كافية. وعلى الرغم من ذلك، فإن النمذجة التمهيدية جديرة بالاعتبار عند تطوير برمجيات المقررات التعليمية؛ حيث تسهم لغات التأليف فى إمكانية ذلك.

7 - الوسائل / الوسائط المتعددة:

ما الاختلاف فى عملية تطوير برمجيات المقررات التعليمية عند استخدام الوسائل / الوسائط المتعددة Multimedia؟ فى الماضى، كان التدريب أو التعليم المبنى على الكمبيوتر Computer-Based Training (CBT) مرتكزا على عرض النصوص الثابتة فقط. بعدئذ أصبح فى الإمكان توظيف وسائل أو وسائط مثل الرسومات، الأنماط المتغيرة، الحركة، الصوت وأخيرا الفيديو. وكل وسيلة من هذه الوسائل فى نمط العرض تحتاج إلى الاختيار والتصميم والإنتاج قبل أن تتكامل مع البرمجيات.

ويأخذ اختيار الوسيلة مكانا فى مستويين: أحدها على مستوى المقرر التعليمى العريض والآخر على مستوى الوحدة أو الموديول التعليمية، حيث يجب الاختيار بين المواد المطبوعة، المحاضرة، التدريس بالكمبيوتر، المحاكاة... الخ. وفى قياس الوسيلة المختارة يجب تأكيد دقة القياس فى إطار درس التعلم بمساعدة الكمبيوتر CAL، حيث تتخذ القرارات عند استخدام الرسومات، الصوت المرتفع، الحركة... الخ، لتوصيل المحتوى التعليمى وكجزء من واجهة التفاعل مع المستخدم. وبالتوازى مع تصميم أوجه برامج المقرر التعليمى الأخرى (وخاصة أنشطة المتعلم وواجهة التفاعل مع المستخدم) يجب أن يصمم أيضا كل نمط عرض جوهرى. على سبيل المثال، يجب أن يصمم محتوى النص والهيكلية والمظهر الخارجى، كما يجب أن يصمم المحتوى وخصائص الفيديو والأوديو التى تحدد كنصوص Scripts.

وبعد التصميم، يجب إخراج كل وسيلة أو وسيط. وفى التعليم أو التدريب المبنى على الكمبيوتر التقليدى تعتبر الوسيلة الوحيدة هى النص ولكنها ما تزال تحتاج إلى التصميم أيضا طبقا للمبادئ التعليمية والمطبعية ومع توجيهات للعرض على الشاشة (Clarke, 1992).

وغالبا يكتسب كل من المؤلفين والمصممين التعليميين هذه المهارات. وتعتبر الوسائل الأخرى أكثر تعقيدا وتتطلب مهارات متخصصة لإنتاج أشكال واقعية للصور الفوتوغرافية، الحركة، الفيديو، والأوديو... الخ. وعندما لا يتوافر فريق عمل كبير نسبيا تنتج برامج المقررات التعليمية فى بيئات تجارية، حيث إنها تشترى من الاستوديوهات لمجابهة توصيف التصميم.

وبعد إنتاج كل وسيلة بطريقة تتسم بالاستقلال، يجب تكامل كل الوسائل معا فى رقابة البرمجيات وفى واجهة تفاعلها مع المستخدم.

هل يؤثر استخدام الوسائل / الوسائط الإضافية على دورة حياة عملية التطوير الشاملة؟، للإجابة على هذا السؤال قام "مارشال وآخرون" (Marshall, et al, 1994) بمقارنة دورة حياة عملية التطوير الخطى العامة التقليدية Waterfall مع نموذج دورة حياة تطوير برمجيات المقررات التعليمية التى تستخدم الوسائل / الوسائط المتعددة كما فى الشكلين رقمى (1/8) و (2/8) السابقين.

ويلاحظ أن مرحلة التصميم تنقسم إلى نشاطين أو مرحلتين: أحدهما للتصميم التعليمى الشامل، والأخرى لتصميم الوسائل المفصل، كما أن مرحلة الإنتاج تنقسم أيضا إلى إنتاج الوسائل / الوسائط وتكامل برامج المقرر التعليمى وربط كل الوسائل معا مع رقابة البرمجيات حتى تكون جاهزة للاختبار. ويصبح الإنتاج أكثر تعقيدا كلما زاد عدد الوسائل التى تتكامل معا.

وبعد مرحلتى التحليل والتصميم يجزئ "ساندفورد" (Sandford, 1990) مرحلة الإنتاج إلى مرحلتين: الجزء الأول لإنتاج كل وسيلة ورقابة البرمجيات اللتين

تطوران بالتتابع أو بالتوازى ما دام تصميم إنتاج مستقلا. والجزء الثانى يرتبط بتكامل الوسائل فى حزمة واحدة. ويعنى ذلك إمداد كل وسيلة منتهية فى شكل مقروء آليا بواسطة البرمجيات الرقابية التى تتبع بتعديل دقيق لاستخدامها وتفاعلها. على سبيل المثال، تهيئة وضع الرسومات أو نقاط التوقف وقت قراءة النص وتحسين التطابق خلال الوسائل.

ويوضح "فوغن" (Vaughn, 1994) فى كتابه عن الوسائل المتعددة مخطط تطوير موسع على الرغم من أنه يتضمن خطوتين فقط قبل إنتاج الأوديو والفيديو وبعد إنتاجهما. ويهتم معظم المخططين بأوجه التصميم الذى يطبق نص برامج المقرر التعليمى فقط، ولكن كل خطوة تعمل بطريقة أكثر تعقيدا، وكلما أضيفت وسائل كلما زاد الوقت والموارد الكلية المتضمنة. كما أن التركيب أو النشر يعتبران أيضا أكثر تعقيدا بسبب حجم البيانات والطلب الكبير على المنظمة الخاصة بذلك.

واقترح "كوبر" (Koper, 1995) طريقة تطوير البرمجيات أطلق عليها (PROFIL) موجه فى الأساس لتطوير برامج المقرر التعليمى للوسائل المتعددة الذى يجمع كثيرًا من الأفكار السابق الإشارة إليها. وتحاول هذه الطريقة إلى تكامل كل من التصميم التعليمى، طرق هندسة البرمجيات، النمذجة التمهيدية واختيار الوسائل معا. وتشتمل هذه الطريقة على ستة مراحل متعاقبة، كما تقيد التكرار فى نطاق المراحل بدلا من اكتمال الدائرية Looping بين المراحل المتجاورة أو المتلاحمة معا كما فى نموذج دورة الحياة الخطية التقليدية Waterfall. والمراحل الستة لهذه الطريقة هى:

1. **مرحلة الدراسة التمهيدية** Preliminary Investigation: تنتج خطة العمل (التحليل).
2. **مرحلة التفسير** Definition: تنتج خطة مشروع لكل وسيلة (تصميم تعليمى وتصميم الوسائل).

3. **مرحلة كتابة النص** Script Phase: تنتج نص لكل وسيلة، وتصميم مفصل كاف لمنتجى الوسائل (التصميم المفصل).
4. **مرحلة** التحقق الفنى Technical Realization Phase : تنتج برنامج رئيسى يشتمل على الوسائل ويتكامل مع البرمجيات. والإصدارة التمهيدية Alpha Version تمثل مراجعة الزملاء، أما الإصدارة التجريبية Beta Version فترتبط بتجريب مختبرى مع الطلاب المستهدفين (الإنتاج).
5. **مرحلة التنفيذ** Implementation Phase : تنتج منتج قابل للتركيب (التركيب والنشر).
6. **مرحلة الاستكشاف** Exploitation Phase: تنتج تقييم شامل للبرمجيات (الصيانة).

وبافتراض استخدام أداة تأليف لإنتاج برمجيات الوسائل المتعددة، حدد "باركر" (Barker, 1987) مجموعة من الخطوات لنموذج إنتاج البرمجيات التعليمية الخطية Waterfall للمشروعات التعليمية الصغيرة والمتوسطة. وفيما يلى مجموعة من الأسئلة التى تمثل الإجابة عليها تنفيذ خطوات النموذج:

1. هل توجد حاجة لإنتاج البرمجيات؟ (متطلبات التعليم أو التدريب).
2. من يتعلم؟ (تحديد قدرات المتعلمين، الطلاب، المتدربين).
3. ماذا يدرس؟ (تحليل المهمة).
4. ما مستوى التعليم المحتاج إليه؟ (التعمق والتفاصيل).
5. كيف تنظم المادة المطلوبة للتعليم؟ (تصميم الدرس، طرق التدريس، الاختبارات).
6. ما الموارد المطلوبة للاستخدام؟ (حاسبات آلية، النصوص، الفيديو).
7. كيف تقوم فعالية البرنامج؟ (التجريب).
8. ما أوجه مراجعة البرنامج؟ (التقييم والصيانة).

كما وضح "شويل" (Shuell, 1992) هذا النموذج التتابعى الخطى Waterfall فى إنتاج برمجيات الوسائل المتعددة فى القيام بالأنشطة التالية:

1. تعريف الأغراض والأهداف.
2. اعتبار المستخدم (المتعلم / الطالب).
3. تحديد الإجراءات التعليمية المتمثلة فى:
   - عرض المعرفة المطلوب تزويد الطالب بها.
   - حث وإثارة المتعلم على التعلم.
   - تضمين العمليات النفسية "السيكولوجية" الضرورية.
4. تقييم معرفة وفهم المتعلم.
5. التقديم للتعليم البديل (إعادة العلاج والتكيف مع الاختلافات الفردية).
6. اختبار ميدانى تجريبى مع طلاب حقيقيين وعمل التغييرات كلما كان ذلك ضروريا.

وقدم "جوناسم" (Jonassem, 1988:3) نموذجا آخر يتسم بالرسمية، وضح فيه المراحل التالية:

1. **مرحلة التحليل**: Analysis Phase الذى يختص بالأنشطة التالية:
   - تعريف المشكلات التعليمية: تقييم الحاجات / تحليل الأداء.
   - تحليل المهمة التعليمية: اختيار محتوى المهمة، الوصف، التتابع.
   - تطوير أهداف الأداء.
   - اختيار استراتيجيات الإمداد أو الإتاحة.
2. **مرحلة التطوير / التجميع** Development / Synthesis Phase وتشتمل على الأنشطة التالية:
   - تقرير استراتيجيات الإمداد أو الإتاحة.
   - اختيار الوسائل والمواد.

- تطوير المواد غير المباشرة والخارجة عن الخط Outline، لوحة العرض Storyboard، نموذج الشفرة التمهيدية Code Prototype... الخ.
- تطوير الاستراتيجيات الإدارية.

3. **مرحلة التقييم** Evaluation Phase وتتضمن التالى:
- تقدير وتقييم مهارات الدخول والاستخدام.
- إدارة وتقييم التعلم ومراجعة النموذج التمهيدى.
- إنتاج وتوزيع وتنفيذ التعليم.
- تقييم النظام.

ويوضح "جوديير" (Goodyear, 1994) فى عمله عن الأساسيات لهندسة البرمجيات التعليمية سبع مراحل لنموذج التطوير الذى اقترحه، وهى:

1. توصيف المتطلبات.
2. التوصيف الوظيفى.
3. التصميم.
4. التنفيذ.
5. التكامل.
6. المراجعة والصحة.
7. الصيانة.

ومن قبل، حدد "روبلاير" (Roblyer, 1988) ثلاث مراحل أساسية لتطوير البرامج التعليمية وفقا للتالى:

المرحلة الأولى: التصميم: وتتضمن الأنشطة التالية:
- تحديد الهدف التعليمى.
- أداء التحليل التعليمى.
- تطوير أهداف الأداء.
- تطوير استراتيجيات الاختبارات.
- تصميم الاستراتيجيات التعليمية.

المرحلة الثانية: تطوير ما قبل البرمجة: وتشتمل أنشطة ما قبل البرمجة Pre-Programming على:

- تطوير خرائط التدفق Flowcharts ولوحات العرض Storyboard .
- تطوير المواد المساندة.
- التصميم ومراجعة فريق العمل.

المرحلة الثالثة: التطوير / التقييم: ويختص ذلك بالتالى:

- مواد مسودة البرنامج الأولية.
- أداء التقييم التشكيلى Formative (مع كل مرحلة توجد تغذية مرتدة Feedback للمراجعة).

كما قدم "اسبكتور وآخرون" (Spector et al, 1992) خمس مراحل أساسية لمخطط تطوير برمجيات الوسائل المتعددة ينمثل فى التالى:

1. **التحليل**:

- تفسير متطلبات التعليم أو التدريب.
- تحليل سمات الطلاب أو المتدربين المستهدفين.
- تحديد وإنشاء مستويات الأداء.

2. **التصميم**:

- تحديد الأهداف التعليمية.
- تحديد أهداف المقرر أو المادة التعليمية والموديولات التابعة.
- تصميم التعاملات التعليمية.

3. **الإنتاج**:

- تطوير أنشطة التعلم.
- تطوير وحدات الاختبار.
- تطوير النماذج التمهيدية Prototypes.

4. **التنفيذ**:

- تنفيذ أنشطة التعلم.
- إدارة وحدات الاختبار.

- تقييم نتائج الطلاب.

5. **الصيانة:**

- مراجعة مواد المقرر التعليمى.
- مراجعة وحدات الاختبار.
- تقييم فعالية المقرر التعليمى.

ومن النماذج أو المخططات العملية لهندسة برامج المقررات التعليمية النموذج أو المخطط التالى:

1. دراسة الجدوى Feasibility Study [التحليل]

↓ التقرير النهائى

2. تحليل الحاجات Needs Analysis

↓ توصيف المتطلبات

3. مرحلة التصميم Design Phase

↓ توصيف الوظائف

4. مرحلة الإنتاج Production Phase

↓ إنتاج الموديول + النموذج التمهيدي

5. دورة النمذجة التمهيدية Prototyping Circle

↓ الاختبارات

6. الاختبار التجريبى والميدانى والمراجعة Pilot Field Testing and Revision

↓

7. التركيب Installation

شكل رقم (7/8): نموذج تطبيقى لهندسة برمجيات المقررات التعليمية

يتضح من المخططات أو النماذج السابقة أن معظمها يشترك فى التتابع العام لمراحل التحليل، التصميم، الإنتاج والتركيب. ويصاحب ذلك التقويم والمراجعة المستمرة فى معظم المراحل. وبينما تستخدم طريقة النمذجة التمهيدية فى كل المخططات يمكن ملاحظة أنها تختلف عن طريقة دورة حياة التطوير المتتابعة

Waterfall التى ذكرت فى بعض الخطط والنماذج، وتتضمن معالم الاختلافات فى التالى:

- عند افتراض أن المتطلبات التمهيدية تحدد بوضوح كاف؛ فإنه بناء على ذلك لا تصبح مرحلتا دراسة الجدوى والتحليل جزءا من دورة التصميم.
- باختلاف دورة التصميم الأولية وإنتاج نموذج تمهيدي؛ فإن أجزاء الوسائل المنفصلة تصمم ويعد لها النص Script ولا تراجع فيما بعد.
- تصميم التركيب المعيارى Modular لبرامج المقرر التعليمى يعنى أن الموديول الأول يحتاج مراجعة كبيرة، بينما تبنى عليها الموديولات الأخيرة.

باختصار، يحتاج تطوير الوسائل المتعددة التفاعلية كما فى برمجيات المقررات التعليمية والوسائل الإضافية المصاحبة لها أن تصمم وتنتج معا بالتوازى مع رقابة البرمجيات. ويشتمل الإنتاج على تكامل الوسائل معا فى تفاعلها مع المستخدم.

8 - أدوات البرمجيات:

تماما كما فى هندسة البرمجيات، يمكن أن تستخدم أدوات البرمجيات فى طرق كثيرة ترتبط بعملية تطوير برمجيات المقررات التعليمية. وتشتمل أدوات هندسة برمجيات المقررات التعليمية بمساعدة الكمبيوتر التى حددت من قبل كل من "هارمون ، وهول"(Harmon & Hall, 1993) على مجموعة الأدوات التالية:

- أدوات إنتاجية عضوية Generic لإدارة المشروع.
- لغات برمجة الجيل الرابع 4th Generation Languages مثل أدوات التأليف للنماذج التمهيدية والمنتج النهائى.
- هندسة البرمجيات بمساعدة الكمبيوتر فى مرحلة أدنى Lower CASE؛ حيث تنتج بعض أدوات التأليف كالشفرة أو التكويد.
- هندسة البرمجيات بمساعدة الكمبيوتر فى مرحلة أعلى Upper Level التى تساند أدوات تحليل وتصميم عملية تطوير البرمجيات.

- الأدوات المتكاملة الآلية لإنتاج برامج المقرر التعليمى من حيث إمكانية آلية هندسة برامج المقرر التعليمى.

ويمكن وصف مجموعات الأدوات فى التالى:

(1) **إدارة المشروع: Project Management يمكن أن تساعد أدوات الإنتاجية الشخصية فى تطوير برامج المقررات التعليمية بطرق واضحة كثيرة، وفى هذه الحالة يمكن استخدام برمجيات معالجة النص Word Processing** ، برمجيات أداة تأليف المجال Domain Authoring Tool ، والجداول الإلكترونية Spreadsheets ، وبرمجيات نظم إدارة قاعدة البيانات DBMS ، وبرمجيات إدارة المشروع Project Management لتحديد تكلفة وتصميم نماذج المحاكاة وقواعد البيانات لمتابعة الموارد المستخدمة. وكلما أنتجت البرمجيات التعليمية بواسطة فرق عمل أو مجموعات البرمجيات Groupware على شبكة الحاسبات أو على الإنترنت، فإن ذلك يسرع عملية تكاملها. وتشتمل أدوات العصف الذهنى Brainstorming على لوحات العرض أو السبورات الإلكترونية e-Whiteboard التى توفر نسخا مطبوعة Hardcopy. ويمكن استخدام برمجيات إدارة المشروع للرقابة على المشروعات من كل الأنواع المتضمنة فى تطوير برمجيات المقررات التعليمية.

(2) **هندسة البرمجيات بمساعدة الكمبيوتر ذات مستوى أعلى : Upper CASE** تساعد هذه البرمجيات فى تحليل نظم المقررات التعليمية. وتشتمل الأدوات التى تساند تحليل المحتوى التعليمى كما وضحه كل من "آلسى وترولیب" (Allessi & Ttrolip, 1991) فيما يتعلق بالوظائف فى معالجة النص التى تنتج هرمية من الأفكار. وتوجد حزم برمجيات لرسم الشبكات مثل الشبكات الدلالية Semantic Network، كما توجد أيضا أدلة لمقابلة خبراء الموضوع عند استنباط المعرفة.

(3) **أداة تأليف المجال: Domain Authoring Tool (DAT)** تساعد هذه الأداة فى

نمذجة المعرفة المساندة لتصميم النظم التعليمية. وتنجز هذه الأداة أربعة وظائف هى:

- تحليل المتطلبات: Requirements Analysis تسجل المعلومات عن المتعلم، الطالب أو المتدرب قبل تعريضه للمعرفة، وتحديد نمط المتعلم وعوامل حثه وتحفيزه، واسم المجال، وغرض المهام المنجزة بعد عملية التعليم.
- تحليل المجال: Domain Analysis تتطلب النمذجة كشبكة أجزاء أو أقسام المجال، الأقسام الفرعية والتفاعلات فى الهرمية. ويستحيل إضافة وصلات هرمية فيما بعد، ويمثل ذلك تجزيء الموضوع بفعالية لإنتاج شبكة دلالية.
- تحليل غرض التعلم: Learning Goal Analysis يرتبط ذلك بمهام غرض التعلم الشمولية ويجزئها مع أخذ تحليل المجال فى الحسبان. ومنتج ذلك يتمثل فى قائمة مهارات ومعرفة هيكلية.
- تحليل المحتوى: Content Analysis يحول تحليل الغرض فى محتوى التعلم. ويتمثل المنتج فى قائمة أو شبكة من النصوص تحدد ما يجب تعليمه (كالوحدات أو الموديولات وعلاقاتها)، ولكن لا تحدد كيف يجب أن تعلم الوحدات. وتعتبر أداة تأليف المجال Domain Authoring Tool أداة سلبية ساكنة، حيث إن الخبرة الموضوعية تأتى من إبداع أو تأليف المؤلف. ويمكن النظر إلى هذه الأداة بأنها تمثل أداة رسم Template مقيدة، لأنها تسهم فى تنظيم وصف المجال وتجزيء أغراض أو أهداف التعلم. وبذلك تعتبر هذه الأداة بأنها تحليلية، إلا أن تحويل تحليل هدف التعلم فى تحليل المحتوى الذى يبدأ من مرحلة التصميم. وتتضمن هذه الأداة مزايا إنتاجية عامة لأداة البرمجيات تختلف عن الطرق التقليدية المعتمدة على (المراجعة والتجزيء) وتتفق مع الفحص والتكامل للمؤلف، ولكنها تتخلف عنها بسبب صعوبة عملية تقرير كيف يدرس المحتوى المعرفى.

(4) **أدوات التصميم: Design Tools** تعتبر بعض أدوات التصميم بسيطة مثل خرائط التدفق أو خرائط سريان الإجراءات Flowcharting، وتشتمل أيضا على أدوات ترتبط بالإنتاجية من إعادة رسم المسودات على الورق. ويمكن تصميم الشاشات مع أداة أو نوع أداة العرض الأخرى مثل البرمجيات الحرة Freelance Software أو برمجيات Power Point Software.

(5) **الأدوات المتكاملة والآلية: Integrated Tools and Automation** يقترح التكامل التحرك ببساطة وسلاسة من التحليل إلى التصميم حتى إلى الإنتاج. وتعتبر ترجمة الاحتياجات المحللة إلى الوظائف المصممة أصعب جزء من التطوير التعليمى. وتستخدم فى هذا الصدد طرق مختلفة مثل طريقة "جاجنيه Gagne" المرتبطة بطريقة التعلم الظاهرى كإطار لتلك الترجمة. وتعتبر آلية التطوير التعليمى للمقررات التعليمية أكثر صعوبة من أدوات هندسة البرمجيات بمساعدة الكمبيوتر CASE، لأن النموذج المحدد لها يرتبط بنظرية التعلم البشرى. ويميز "جوديير" (Goodyear, 1995) أدوات المساندة أو الدعم الضعيفة أو القوية على حد سواء. ويعتبر الدعم الضعيف دعما ساكنا وسلبيا على الرغم من أنه يريح ويقلل جهد المحلل أو الجهد المعرفى ليسمح لمبرمج برمجيات المقررات التعليمية Courseware التركيز على أوجه أخرى أكثر أهمية. أما الأدوات القوية فإنها تمكن الاستخدام الآلى لبعض القرارات، وتحتاج إلى أن تبنى على نموذج تعلم وتتضمن خبرة عالية. وتتضمن الأدوات القوية فكرتين أساسيتين: أحداهما تبنى على نظرية تعلم وتعليم معينة وتسمح هيكليتها بمداخل معينة للتصميم التعليمى. ويشبه ذلك رسم قياسى Template مفصل، كما تشبه أيضا أدوات هندسة البرمجيات بمساعدة الكمبيوتر CASE . أما الفكرة الثانية الأخرى فتعتبر أضعف وتمثل مدخلا تجريبيا إلى حد ما لتضمين نظام خبرة Expert System يقترح قرارات تصميم معينة. ويصف كل من "ويلسون وجوناسن" (Wilson and Jonassen, 1991) إدخال الآلية على بعض أجزاء عملية التطوير.

وباختصار يوجد تنوع من أدوات تطوير برمجيات التى تساعد فى تطوير برمجيات المقررات التعليمية تشتمل فيها الأدوات الضعيفة (الساكنة أو السلبية) على مساندة لإدارة المشروع، التحليل والتصميم. وتستخدم أدوات التأليف فى إنتاج وتغيير نموذج التطوير. وتعمل أدوات التطوير القوية على آلية التصميم ولكنها ما زالت فى بداياتها غير الناضجة كليا.

9 - الخلاصة:

يستنتج من العرض السابق أن هندسة برمجيات المقررات التعليمية Courseware تعتمد على كل من التصميم التعليمى وهندسة البرمجيات كمجالين متكاملين معا فى عملية التطوير.

ويساند التصميم التعليمى عملية التعلم والتدريس فيما يتصل بتحديد قدرات وملكات المتعلمين، أهداف التعلم، والمحتوى التعليمى وتتابعه واختباراته التى ترتبط بنماذج استراتيجيات التعلم، سواء كانت نحو التعلم البنائى، الفردى، الجماعى، المجتمعى، معالجة المعلومات، حل المشكلات أو صقل المهارات.

وتتعدى هندسة البرمجيات البرمجة الحرفية المنظمة البحتة بحيث ترتبط بعلم الحاسب الآلى المعتمد على أساس نظرى متكامل، كما تفرض نظما معينة دقيقة ومتقنة تتصل بأداء التطبيق الكمبيوترى مما يمثل مدخلا منظما لتطوير وتشغيل وصيانة وإدارة البرمجيات المطورة إلى أن تتوقف نظرا لتقادمها أو لظهور أدوات أحدث وأقوى فى إعادة هندستها.

أما هندسة برمجيات المقررات التعليمية فإنها تمثل مجموعة من المزاولات والأدوات والمنهجيات التى تطبق المدخل الهندسى المرتبط بالدقة والإتقان الذى يستخدم طرقا وأدوات معيارية مقننة على عملية التطوير بدلا من الحدس والتخمين فقط. ويطبق ذلك على التحليل والتصميم والإنتاج وإتاحة البرمجيات ووضعها موضع التنفيذ والإدارة.

وتوجد عدة نماذج من عملية تطوير برمجيات المقررات التعليمية متاحة ومنشورة بالفعل. وقد تتنوع هذه النماذج وتختلف عن بعضها البعض، لا بسبب النظريات التعليمية المشكلة لها فى إطار التصميم التعليمى، ولكن بسبب تطبيق طرق وأدوات مختلفة فى ظروف متنوعة . إلا أن معظم هذه النماذج المتاحة ترتبط فى الأساس بمراحل التحليل والتصميم والإنتاج والتقييم والمراجعة والإدارة.

وتتصل دورة حياة عملية تطوير نظم وبرمجيات المقررات التعليمية إما بالمدخل التقليدى الخطى والتتابعى الذى يطلق عليه Waterfall المرتبط بتتابع المراحل والمتسم بالبطء والتكلفة العالية، أو بمدخل النمذجة التمهيدية Prototyping المتسم بالمرونة والسرعة والاقتصاد فى التكلفة والمتوافق مع أدوات الجيل الرابع لبرمجة المقررات التعليمية التى قد توظف تكنولوجيا الوسائط المتعدد التفاعلية.

المراجع

1. Allessi, S. M. and Trollip, S. R. (1991). Computer-based instruction. 2nd ed. Englewood-Cliffs, NJ: Prentice-Hall,
2. Barker, P. (1987). Authoring languages. London: Croom Helm
3. Barron, A. E. et al (1995). "A model of interaction: in search of Holy Grail" In: Tennyson, R. D. and Barron, A. E. (eds.) Automating instructional design: computer based development and delivery tools. Berlin: Springer-Verlag (Chapter 24).
4. Clarke, A. (1992). The Principles of screen design for computer based learning materials. 2nd ed. Washington, DC: Employment Department Group
5. Dean, Cristopher and Whitlock, Quentin (1992). Handbook of CBT. 2nd ed. London: Kogan Page
6. De Diana, I. and van Schaik, P. (1993). Courseware engineering outlined: an overview of some research issues. **ETTI** , Vol. 30, N0. 3, pp. 191-211.
7. Foster, G. (1993). "Managing course design", **British Journal of Educational Technology**, Vol. 24, N0. 3, pp. 198-20.
8. Friedler, Y. and Shabo, A. (1991). "An Approach to cost-effective courseware development". **British Journal of Educational Technology**, Vol. 22, N0. 2, pp. 129-138.
9. Gagne, R. M. (1985). The condition of learning. 4th ed.
10. Goodyear, peter (1994). Foundations for courseware engineering. In: Tennyson, R. D. and Barron, A. E. (eds.) Automating instructional design, op. cit.
11. Goodyear, R. M. (1994). "Infrastructure for courseware engineering" In: Tennyson, R. D. and Barron, A. E. (eds.) Automating instructional design: computer based development and delivery tools. op. Cit.
12. Harmon, P. and Hall, C. (1993). "Intelligent software systems development". In: Case Technology. New York: John Wiley (chapter 8).
13. Jonassen, D. H. (1988). "Instructional design and courseware design". In: Jonassen, D. H. (ed.). Instructional designs for microcomputer courseware, LEA.
14. Koper, R. (1995). PROFIL: a method for the development of multimedia courseware. **British Journal of Educational Technology**, Vol. 26, N0. 2, pp. 94-108.
15. Petry, Mouton and Reigluth, C. M. "A lesson based on Gagne-Briggs theory of instruction". In: Reigluth, C. M. (ed.) Instructional theories in action (Chapter 2).

16. Roblyer, M. D. (1988). "Fundamental problems and principles of designing effective courseware". In: Jonassen, D. H. (ed.) Instructional designs for microcomputer courseware, op. cit.

17. Sandford, N. (1990). Keeping alligators under control: the benefits of visualizing models and other prototyping methods in early evaluation, **Education and Training Technology International**, Vol. 27, N0. 2, pp. 174-182.

18. Schach, S. R. (1990). Software engineering. Homewood, IL: Aksen Associates.

19. Sommerville, I. (1993) Software engineering. 3rd ed. New York: Addison Wesley.

20. Shuell, T. J. (1992). "Designing instructional computing systems for meaningful learning", In: Jones, M. and Winne, P. H. (eds.) Adaptive learning environments. Berlin: Springer-Verlag.

21. Spector, J. M., Gagne, R. M., Muraida, D. J. and Dimitroff, W. A. (1992)." Intelligent frameworks for instructional design", **Educational Technology**, pp. 21-27.

22. Tripp, S. D. and Bichwelmeyer, B. (1990). "Rapid prototyping: an alternative instructional design". **Educational Technology Research and Development**, Vo. 38, N0. 3, pp. 31-43.

23. Vaughn, T, (1994). Multimedia: making it work. 2nd ed. Osborne: McGraw-Hill.

24. Van Vliet, H, (1993). Software engineering. New Yorkr: John Wiley.

25. Wilson, H. amd Jonassen, D. (1991). "Automated instructional design: a review of prototype systems". **Journal of Artificial Intelligence in Education**, Vol. 2, N0. 2, pp. 17-30.

26. Witt, C. L. and Wager, W. (1994). "A comparison of instructional systems design and electronic performance systems design". **Educational Technology**, pp. 20-24.